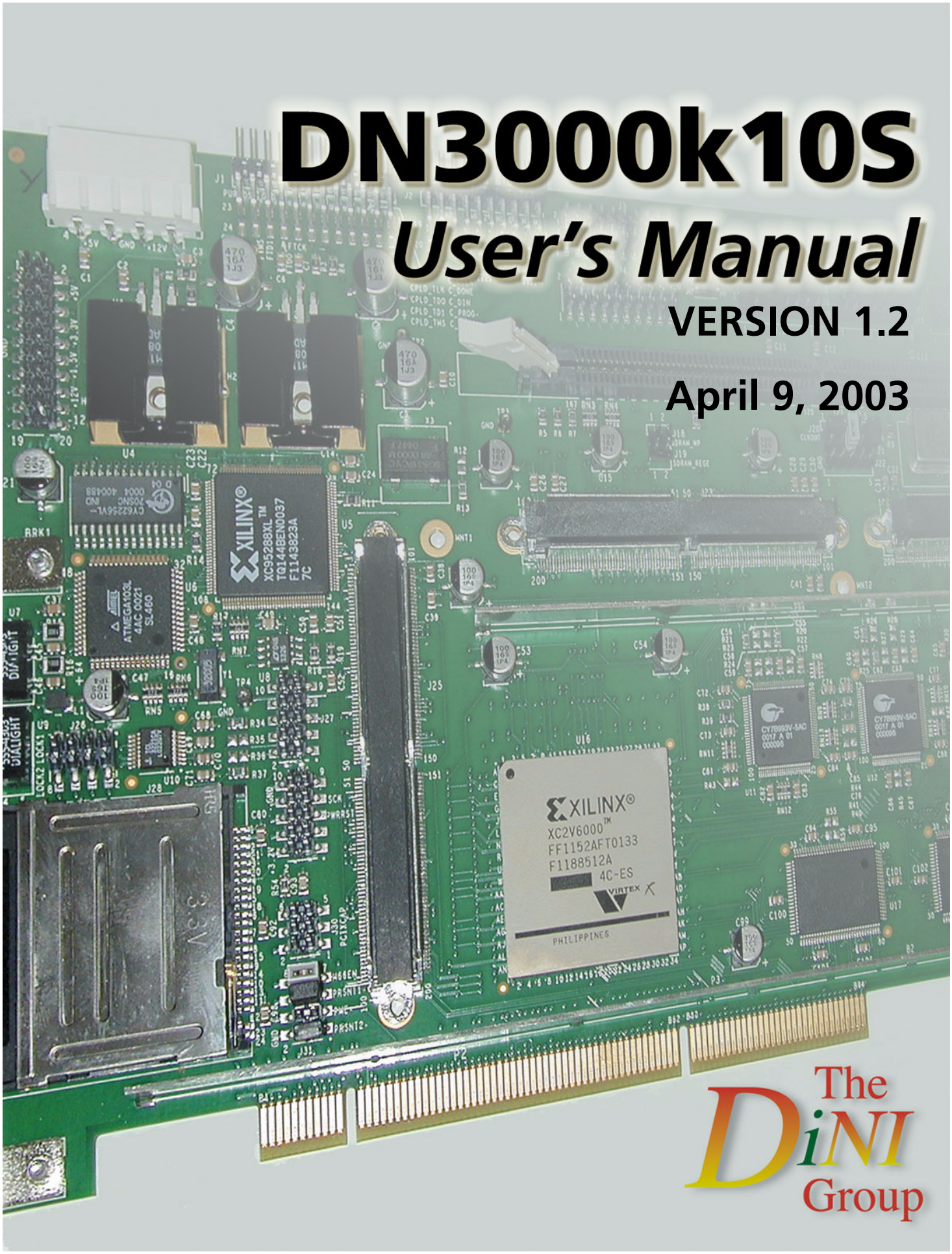
DN3000k10S

User's Manual

VERSION 1.2

April 9, 2003



The DiNI Group

DN3000k10S User's Manual Version 1.2

April 9, 2003

.

1010 Pearl Street, Suite #6

La Jolla, CA 92037-5165

www.dinigroup.com

info@dinigroup.com

(858) 454-3419

FAX: (858) 454-1728

Table of Contents

List of Figures

List of Figures (Continued)

List of Figures (Continued)

List of Tables

Chapter 1

Getting Started

The DN3000k10S is sensitive to static electricity, so treat the PWB accordingly. The target market for this product is engineers that are familiar with FPGAs and circuit boards, so a lecture in ESD really isn't appropriate (and wouldn't be read anyway). However, we have sold some of these units to people who are not as familiar with this issue. The following web page has an excellent tutorial on the Fundamentals of ESD for those of you who are new to ESD-sensitive products:

http://www.esda.org/basics/part1.cfm.

The DINI Group Technical Support

The following means of technical support are available:

1. **The DN3000k10S User's Manual.** This is the main source of technical information. We strive to produce excellent documentation, and this manual should contain most of the answers to your questions.
2. **The DINI Group Web Page.** The web page will contain the latest manual, application notes, FAQ, articles, and any device errata and manual addenda. Please visit and bookmark:
   http://www.dinigroup.com/index.php?product=DN3000k10s
3. **E-Mail** to **support@dinigroup.com**. You may direct questions and feedback to The DINI Group using this e-mail address.
4. **Phone Support.** We are happy to help. Call us at (858) 454-3419 during the hours of 8:00 A.M. to 5:00 P.M. Pacific Time. Some of us get in early and stay late, so you might try us outside of these hours also.
5. **Frequently Asked Questions**. In the downloads section of our web page you can find a document called **DN3000k10/S Frequently Asked Questions (FAQ)**. We will update this document occasionally with information that may not be in the User's Manual.

Relevant Information

Information about PCI can be obtained from the following sources:

The PCI Special Interest Group has a web page that has lots of good stuff. Copies of the latest PCI specification may be ordered here.

http://www.pcisig.com/
PCI Special Interest Group
2575 NE Kathryn St. #17
Hillsboro, OR 97124
FAX: (503) 693-8344

As of October 2001, the most current versions of the PCI Specifications are:

PCI Local Bus Specification, Revision 2.2
PCI Hot-Plug Specificadtion, Revision 1.0
PCI Power Management Interface Specification, Revision 1.1
PCI-X Addendum to the PCI Local Bus Specification, Revision 1.0a

Other recommended specifications include:

PCIMG 2.0 Compact PCI Specification, Revision 2.1 (or greater)
PCI Industrial Computer Manufacturers Group (PICMG)
401 Edgewater Place, Suite 500
Wakefield, MA 01880, USA
TEL: 781-224-1100
FAX: 781-224-1239
http://www.picmg.org

The best book to get if you need an introduction to PCI is:

PCI System Architecture
Fourth Edition
MindShare, Inc.
Tom Shanley and Don Anderson

Ignore some of the ignorant statements made in the Customer Review section at **http://www.amazon.com/**. This is an excellent book for PCI and well worth the money.

The best book to get if you need an introduction to PCI-X is:

PCI-X System Architecture
MindShare, Inc.
Tom Shanely and Karen Gettman

You are going to need to know Verilog or VHDL to use the VirtexII FPGA. If you need a reference, we recommend the following book for Verilog:

Verilog HDL: A Guide to Digital Design and Synthesis
Samir Palnitkar
ISBN: 0-13-451675-3

If you are one of those people that actually like VHDL, we feel sorry for you. The following books may be helpful:

Essential VHDL: RTL Synthesis Done Right
Sundar Rajan

The IQ Booster: Improve Your IQ Performance Dramatically
Edwin Breecher

Conventions

This manual uses the following conventions. An example illustrates each convention.

- The term PCI-X will be used generically unless there is a specific instance where PCI applies.
- This design guide generically refers to PCI-X protocol. When the PCI-X HalfBridge core is in PCI mode, PCI protocol will be followed.
- `Courier font` denotes the following items:
  - Signals on PCI Bus side of the PCI-X Interface

    `FRAME_IO` (PCI-X Interface signal name)

    `FRAME#` (PCI-X Bus signal name)
  - Signals within the user application

    `BACK_UP, START`
  - Command line input and output

    ```
    setenv XIL_MAP_LOC_CLOSED
    ```
  - HDL pseudocode

    ```
    assign question = to_be | !to_be;

    assign cannot = have_cake & eat_it;
    ```
  - Design file names

    `pcim_top.v, pcim_top.vhd`
- **`Courier bold`** denotes the following items:
  - Signals on the user side of the LogiCORE PCI-X Interface

    `ADDR_VLD`
  - Menu selections or button presses

    `FILE -> OPEN`
- Italic font denotes the following items:
  - Variables in statements which require user-supplied values

    ngdbuild *design_name*
  - References to other manuals

    See the *Libraries Guide* for more information.

– Emphasis in text

It is not a bug, it is a *feature*.

- Dark shading indicates items that are not supported or reserved:

SDONE_I	**in/out**	**Snoop Done signal. Not Supported.**

- Square brackets "[ ]" indicate an optional entry or a bus index:

ngdbuild [*option_name*] *design_name*

DATA[31:0]

- A vertical or horizontal ellipsis indicates repetitive material that has been omitted.

A B C... X Y Z

- The use of "`fn(SIG1. . . SIGn)`" in an HDL pseudocode fragment should be interpreted as "combinational function of signals SIG1 through SIGn.

SUM = fn(A, B, Cin);

- The prefix "`0x`" or the suffix "`h`" indicate hexadecimal notation.

A read of address `0x00110373` returned `45524943h`.

- A "#" an "_n" or a "–" means the signal is active low

`INT#` is active low.

`fpga_inta_n` is active low.

`SRAMCS–` is active low.

Chapter 2

DN3000k10S Features, Overview and General Description

DN3000k10S Features

The DN3000k10S features include:

- 32/64-bit, +3.3V, PCI/PCI-X-based PWB with one Xilinx VirtexII™ FPGA (FF1152 BGA).
  - Initial availability: XC2V6000 (2V4000 and 2V8000 to follow when available).
- 500k ASIC gates per PWB (with 2V6000 — LSI standard)

Device	Flip-Flops	18x18 Multipliers	Embedded Memory — 18k bit blocks
XC2V4000	46,080	120	120
XC2V6000	67,584	144	144
XC2V8000	93,184	168	168

- On-board battery supports VirtexII Data Encryption Standard (DES) Bitstream Encryption
- Fast/Easy FPGA configuration via standard SmartMedia FLASH card
  - Microprocessor controlled (ATmega128L)
  - RS232 port for configuration/operation status and control
  - Fastest possible configuration speed (via S*electMap*)
  - Partial reconfiguration supported
- 5A on-board linear regulator for +3.3V and +1.5V
  - Standalone operation via separate power connector
  - +3.3V not needed on backplane
- 6 low skew clocks distributed to all FPGA and test connectors:
  - 2 CY7B993/4 RoboclockII PLLs
  - 2 socketed oscillators
  - PCI Clock
  - 1 dividable clock via CPLD
- Direct support for Synplicity's *Certify* TDM interconnect multiplexing.

- Robust observation/debug with ~500 connections for logic analyzer observability or for pattern generator stimulus.
- Status LEDs.
- User-designed daughter PWB for custom circuitry and interfaces.
- Chipscope fully supported via JTAG interface.

Figure 2-1 shows a block diagram of the DN3000k10S.

DN3000k10S Description

The DN3000k10S is a complete logic emulation system that enables ASIC or IP designers a vehicle to prototype logic and memory designs for a fraction of the cost of existing solutions. The DN3000k10S can be hosted in a 32/64-bit PCI/PCI-X slot, or can be used stand-alone device. A single DN3000k10S stuffed with one XC2V6000 can emulate up to 500k gates of logic as measured by LSI. A high I/O-count, 1152-pin, flip-chip BGA package is employed. The FF1152 package has 824 I/Os, which allows for abundant connections to daughter connectors and external memories. A total of ~500 test pins are provided on the top of the PWB via high-density connectors for logic analyzer-based debugging, or for pattern generator stimulus. Custom daughter cards can be mounted to these connectors as a means of interfacing the DN3000k10S to application-specific circuits. A reference 32-bit PCI target design and test bench is provided in Verilog at no additional cost.

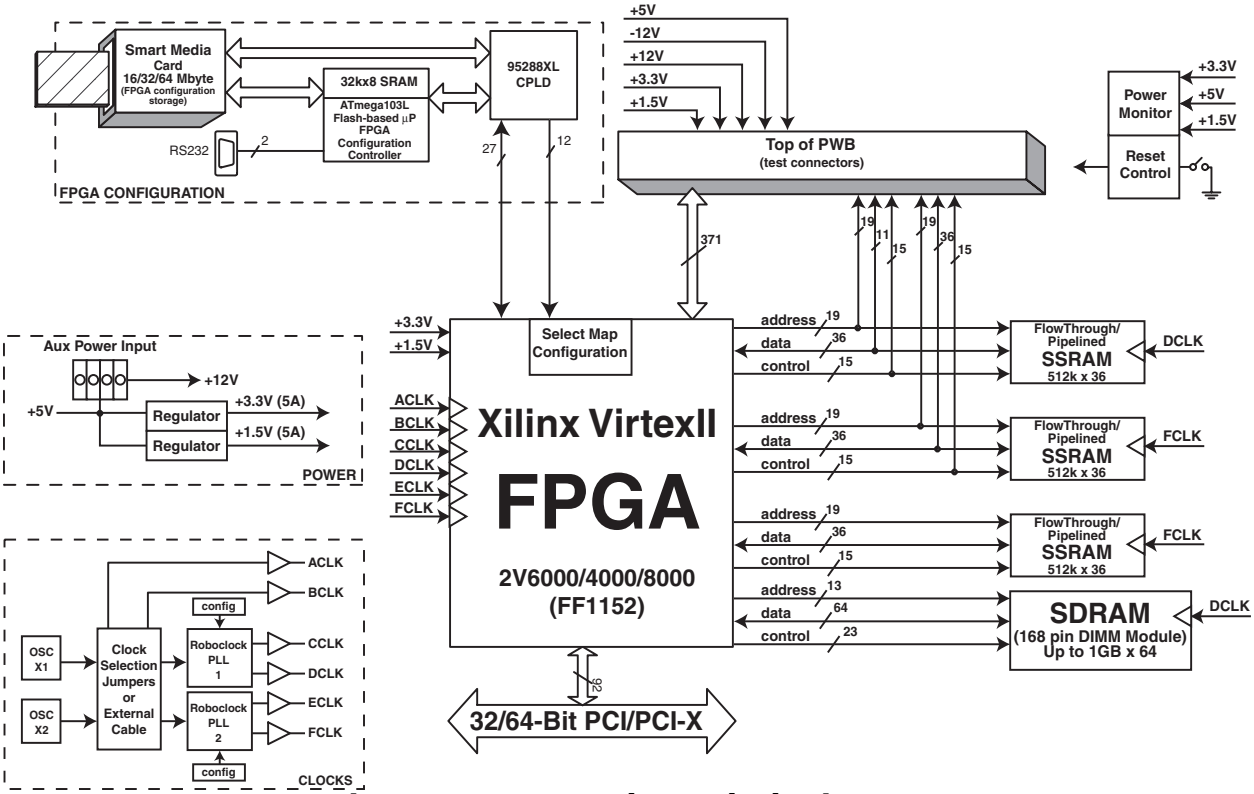


Figure 2-1 DN3000k10S Block Diagram

Easy Configuration via SmartMedia

The configuration bit files for the FPGA are copied onto a 32-megabyte SmartMedia FLASH card (provided) and an on-board microprocessor controls the FPGA configuration process. Visibility into the configuration process is enhanced with an RS232 port. Sanity checks are performed automatically on the configuration bit files, helping to avoid the time-consuming process of debugging the configuration process. FPGA configuration runs quickly at 48 MHz. Eight LEDs provide instant status and operational feedback. Two of these LEDs are connected to the CPLD and can be user-configured.

FPGA — VirtexII (U16)

The DN3000k10S contains a single VirtexII FPGA. The package is a flip-chip fine-pitch BGA with 1152 pins (FF1152). The pitch on the pins is 1 mm. This isn't important, but this pin density makes the PWB a bitch to layout. Keep that in mind if you try to make one of these at home. All 824 I/O pins are utilized on the FF1152 package.

The DN3000k10S can be stuffed with the XC2V4000, XC2V6000, and XC2V8000. The XC2V3000 is available in FF1152, but since it has fewer I/O pins, it should not be used. The standard speed grade we stuff is –4. We can use the –5 and –6 speed grades, but don't fall out of your chair when you get the price. Note that Xilinx has cancelled plans for the XC2V10000. Even though this part appears in some Xilinx literature, we have been told that the XC2V8000 is the largest part that the process will handle. Don't expect to see anything larger than the XC2V8000 until the 2003 time frame.

The following is a very brief overview of the VirtexII family. More information can be gleaned from the VirtexII Datasheet (`VirtexII-Datasheet.pdf`) and the VirtexII User's Guide (`VirtexII-UserGuide.pdf`). Both files are on the CD-ROM supplied with the DN3000k10S, but you are better off getting the latest versions from the Xilinx Web page (**http://www.xilinx.com/**). Make sure to get the latest errata sheet also.

Flip-Flops and LUTs

Figure 2-2 shows what Xilinx calls a *slice*. Each slice contains 2 flip-flops. A configurable logic block (CLB) contains 4 slices. The XC2V6000 is a 96 x 88 grid of CLBs. Therefore, the XC2V6000 contains 67,584 flip-flops. This flip-flop count does not include the six flip-flops contained in each I/O block.

Each flip-flop has a 4 x 1 look up table (LUT). An LUT can do any Boolean function of the four inputs. The rest of the multiplexers allow for carry chains and other functions. An overview of the VirtexII Architecture is shown in Figure 2-3.

Embedded Memory

VirtexII has boatloads of embedded memory. The XC2V6000 contains 144, 18-Kbit blocks. Each memory block can be configured as 16K x 1, 2K x 9, 8K x 2, 1K x 18, 4K x 4 or 512 x 36. Remember that unused LUTs may also be used as memory. Xilinx refers to the embedded memory as *Block SelectRAM*, and to the LUT-based memory as *Distributed Memory*. The

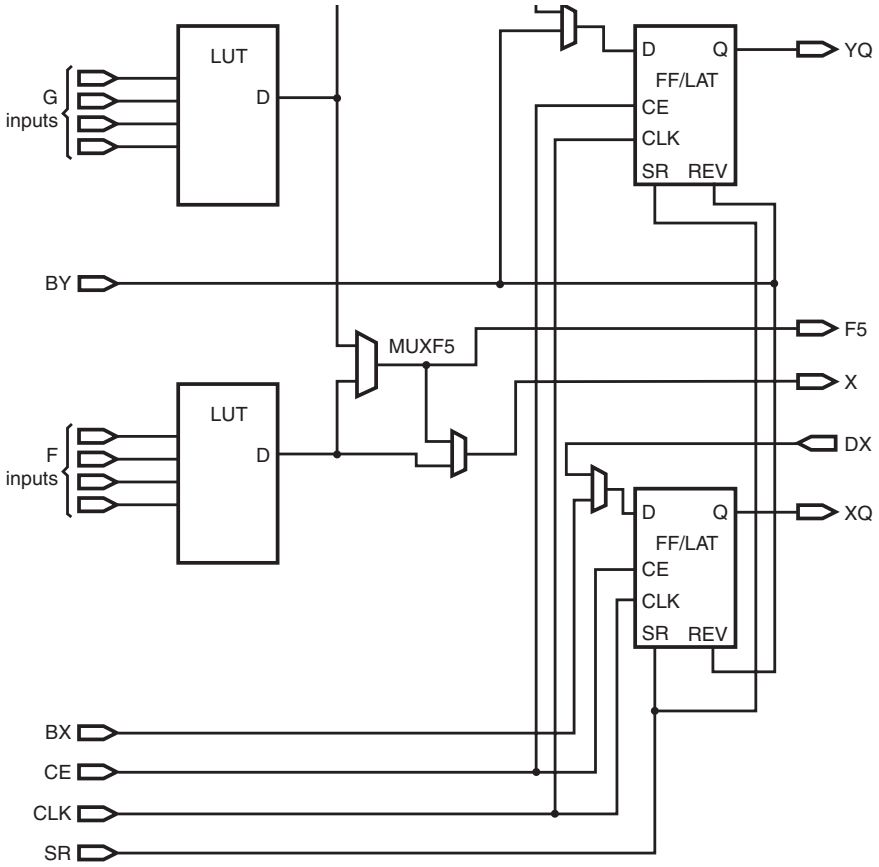


Figure 2-2 General Slice Diagram

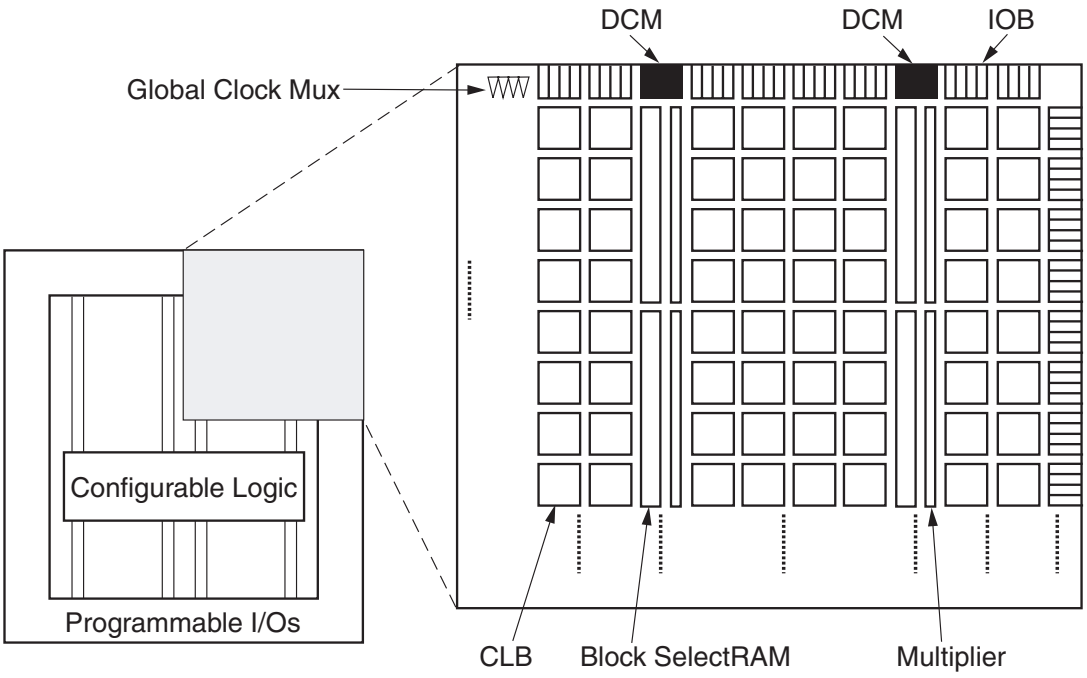


Figure 2-3 VirtexII Architecture Overview

embedded memory is dual-ported and quite flexible. Virtually any type of memory can be constructed—FIFOs,dual-port RAMs, single-port RAMs, etc.

The 18 Kb block SelectRAM dual-port memory consists of an 18 Kb storage area and two completely independent access ports, A and B. The structure is fully symmetrical, and both ports are interchangeable. See Figure 2-4 for a diagram of the memory.

Data can be written to or read from either port in almost any configuration. Each port is synchronous, with its own clock, clock enable, and write enable. Note that the read operation is also synchronous and requires a clock edge.

We have found that a functional description of the memory is sufficient for the synthesis tools to recognize the memories and implement the embedded blocks. More on the synthesis issues in "Synthesis and Emulation Issues" on page 2-29.

Multipliers

VirtexII devices feature a large number of embedded 18-bit x 18-bit two's complement embedded multipliers (see Figure 2-5). The XC2V6000 contains 144 of these; the XC2V4000 contains 120. The embedded multipliers offer fast, efficient means to create 18-bit by 18-bit signed multipliers. The multiplier blocks share routing resources with the Block SelectRAM memory, allowing for increased efficiency for many applications. Cascading of multipliers can be implemented with additional logic resources of VirtexII slices.

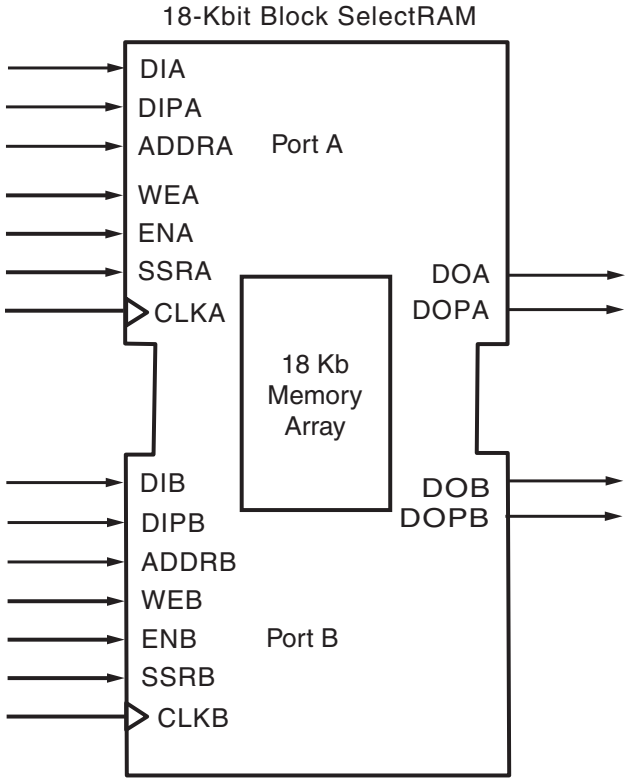


Figure 2-4 Dual-Port Data Flows

Applications such as signed-signed, signed-unsigned, and unsigned-unsigned multiplication, logical, arithmetic, and barrel shifters, two's complement and magnitude return are easily implemented.

We were surprised to find that the synthesis tools had no problem recognizing multipliers in Verilog and VHDL. So it appears that functional RTL is all that is necessary to use the embedded multipliers. More on the synthesis issues in "Synthesis and Emulation Issues" on page 2-29.

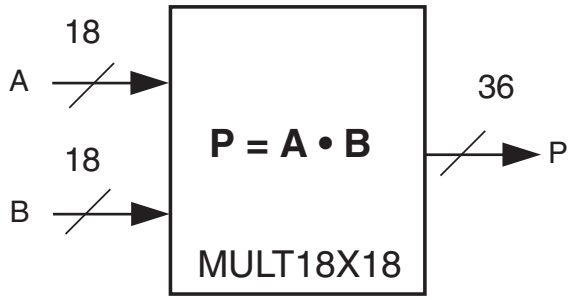


Figure 2-5 Embedded Multiplier

I/O Issues

Digitally Controlled Impedance (DCI) is supported on all pins. The resistors used for VRN and VRP are 51.1 ohms 1%. The PWB impedance is 50 ohms. So the Xilinx tools should be adjusted to reflect the fact that the reference resistors match the board impedance (and **NOT** half the resistance of the reference resistors). DCI is a very nice feature and we recommend you use it on all I/O signals. The correct IOATTRIBUTE standard for the UCF file is **LVDCI_33**.

All VCCO pins are connected to +3.3 V. The VREF pins are used as I/Os, so the DN3000k10S does not support I/O standards that require VREF. No signals have a V_{TT} (Board Termination Voltage). So the I/O standards supported are:

LVTTL — Low-Voltage TTL
The low-voltage TTL, or LVTLL, standard is a general-purpose EIA/JESDSA standard for 3.3 V applications that use the LVTLL input buffer and a Push-Pull output buffer. The standard requires a 3.3 V input and output source voltage (V_{CCO}) but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

LVCMOS33 — 3.3 Volt Low-Voltage CMOS
This standard is an extension of the LVCMOS standard (JESD8. –5). It is used in general-purpose 3.3 V applications. The standard requires a 3.3 V input/output source voltage (V_{CCO}) but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

PCI-X — Peripheral Component Interface
The PCI standard specifies support for 33 MHz, 66 MHz and 133 MHz PCI bus applications. It uses a LVTTL input buffer and a Push-Pull output buffer. This standard does not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}); however, it does require 3.3 V input output source voltage (V_{CCO}).

LVDS is supported on some pins to the 200–pin connectors. See "The 200 Pin Connectors: J23, J24, J25" on page 8–5.

Bitstream Encryptions

VirtexII devices have on-chip decryption circuitry that can be enabled to make the configuration bitstream (and thus the whole logic design) secure. Ultimately, you will be able to encrypt the bitstream in the Xilinx software, and the VirtexII chip will perform the reverse operation, decrypting the incoming bitstream and internally recreating the intended configuration. The DN3000k10S has a battery socket to support this function.

Encrypting the bitstream is important if you want to use the DN3000k10S as a platform to demonstrate intellectual property (IP) but don't want anybody reverse engineering the bitstream in an attempt to steal the design.

NOTE: It is possible to recreate a design from the configuration bitstream!

This method provides a very high degree of design security. Without knowledge of the encryption/decryption key or keys, potential pirates cannot use the externally intercepted bitstream to analyze, or even to clone the design. System manufacturers can be sure that their VirtexII implemented designs cannot be copied and reverse engineered.

The VirtexII devices store the internal decryption keys in a few hundred bits of dedicated RAM, backed up by a small externally-connected battery (**BTI** on the circuit board). At less than100 nA per load, the endurance of the battery is limited only by its shelf life.

The method used to encrypt the data is Data Encryption Standard (DES). This is an official standard supported by the National Institute of Standards and Technology (NIST) and the U.S. Department of Commerce. DES is a symmetric encryption standard that utilizes a 56-bit key. Because of the increased sophistication and speed of today's computing hardware, single DES is no longer considered to be secure. However, the Triple Data Encryption Algorithm (TDEA), otherwise known as triple DES, is authorized for use by U.S. federal organizations to protect sensitive data, and is used by many financial institutions to protect their transactions. Triple DES has yet to be cracked. Both DES and Triple DES are available in VirtexII devices.

What DES Is

DES and Triple DES are symmetric encryption algorithms. This means that the key to encrypt and the key to decrypt are the same. The security of the data is kept by keeping the key secret. This contrasts to a public key system, like RSA or PGP. One thing to note is that VirtexII devices use DES in Cipher Block Chaining mode. This means that each block is combined with the previous encrypted block for added security. DES uses a single 56-bit key to encrypt 64-bit blocks one at a time.

This section is being updated mid-March 2002. At this time, only the Xilinx tools version 4.2 (or a patched 4.1.03) support the encryption function of the Virtex II devices. There are various erratas, updates, and white papers

available at the Xilinx web site (http://www.xilinx.com). Also, our most updated information regarding using Virtex II devices with encryption can be found at our web site at:

http://www.dinigroup.com/products/3000k10ns.html.

As of now, encryption is fully supported using the JTAG chain programming the parts. Encryption using SmartMedia programming will be supported soon. We have found several undocumented features of the Xilinx Virtex II parts with SelectMAP programming that we are now working to accommodate.

Note that encryption and partial reconfiguration are mutually exclusive. You can do one or the other, but not both. There is also a previous errata on the sizes of bit files generated for Virtex II parts being larger than the originally-intended size by approximately 10%. This does **NOT** apply to encrypted bitstreams. The size of an encrypted bit file is the originally-intended size.

For more detailed information regarding Virtex II parts and encryption, see our Encrypted Bitstream documentation at:

http://www.dinigroup.com/products/3000k10ns.html.

Programming Encrypted Bitstreams Using JTAG

1. Creating the Encrypted Bitstreams
   A new option has been added to the Xilinx "`bitgen`" utility for all devices that support encryption. Add the "`-g Encrypt:Yes`" option to the "`bitgen`" command line (see Xilinx User's Guide) to create an encrypted bitstream.

   Two files will be created. The first is the encrypted bit file. the second is a key file, "`.nky`" (which you can specify using the "`-g KeyFile:<filename>`" option). The key file contains the encrypt/decrypt keys used to encrypt/decrypt the bit file.

2. Programming the Key File
   Using the Xilinx iMPACT tool and a JTAG programmer, program the key file "`.nky`" into the FPGA that will decrypt the encrypted bitstreams. Make sure that a +3 V 2032-type lithium coin battery is installed in **BT1** prior to programming. The battery is required for key-retention.

3. Programming the Encrypted Bitstreams
   Using the Xilinx iMPACT tool and a JTAG programmer, program the encrypted bitstreams into the FPGA. The FPGA will then decrypt the bitstream using the previously-programmed decryption keys.

Programming Encrypted Bitstreams Using SmartMedia

The procedure for this is very similar to the procedure for JTAG. However, support for this is not complete and will be coming shortly. For more detailed information, contact us directly or visit our web site at:

http://www.dinigroup.com/products/3000k10ns.html.

The Battery

The DN3000k10S has a socket for a battery, **BT1**. The socket uses a +3 V, 2032 coin-style lithium battery. Don't eat the battery. Most lithium batteries are rated at about 200 mAh, so at 100 nA, the battery should last for 5 years or so.

µP and FPGA Configuration

The DN3000k10S has an ATmega128L microprocessor (µP) that is used to control the configuration process (**U6**). The amount of internal SRAM (4 Kbytes) was not large enough to hold the FAT needed for SmartMedia, so an external 32 k x 8 SRAM was added. The address latching function is done in the XC95288XL CLPD (**U5**).

The microprocessor has the following responsibilities:

- Reading the SmartMedia card
- Configuring the VirtexII FPGA
- Executing DN3000k10S self tests.

Other than FPGA configuration, the µP has no responsibilities. Less than 25% of the 128 Kbytes of FLASH is used for FPGA configuration and utilities, so you are welcome to use the rest of the resources of the µP for your own purposes. Instructions for customizing the µP are contained in the file `Custom_Atmega128L.pdf`. This file is on the CD-ROM, or it can be downloaded from the DINI Group web page.

REMEMBER: You can use the microprocessor for your own purposes!

We ship a programming cable for the ATmega128L with the DN3000k10S. Updates to the code will be posted on our web site. If you wish to do your own development you will need the compiler, which we do not ship with the product. The compiler is available from IAR (**http://www.iar.com/**). The part number is EWA90PCUBLV150.

Note that if you are willing to program the FPGA with the JTAG or serial cable, the CLPD and the µP have **no** function. In this case you can use all of the resources of the µP for your own purposes.

The µP: Some Details

The ATmega128L is gross overkill for the FPGA configuration function. The datasheet and user's manual are on the CD-ROM that was shipped with the DN3000k10S. The file names are `ATmega128_UM.pdf` and `atmega128_DS.pdf`. But if you intend to use the µP for your own purposes, you should check the Atmel web page to get a copy of the latest user's manual, datasheet, and erratas. The Atmel web page is **http://www.eu.atmel.com/atmel/**. The ATmega128L is under the section called "Flash Microcontroller, AVR 8-Bit RISC." Most of the features are unused. A variety of test headers allow for possible use of these features. Each header and the various possible functions are described in the sections that follow. Figure 2-6 is a block diagram of the ATmega128L and its various interfaces on the DN3000k10S.

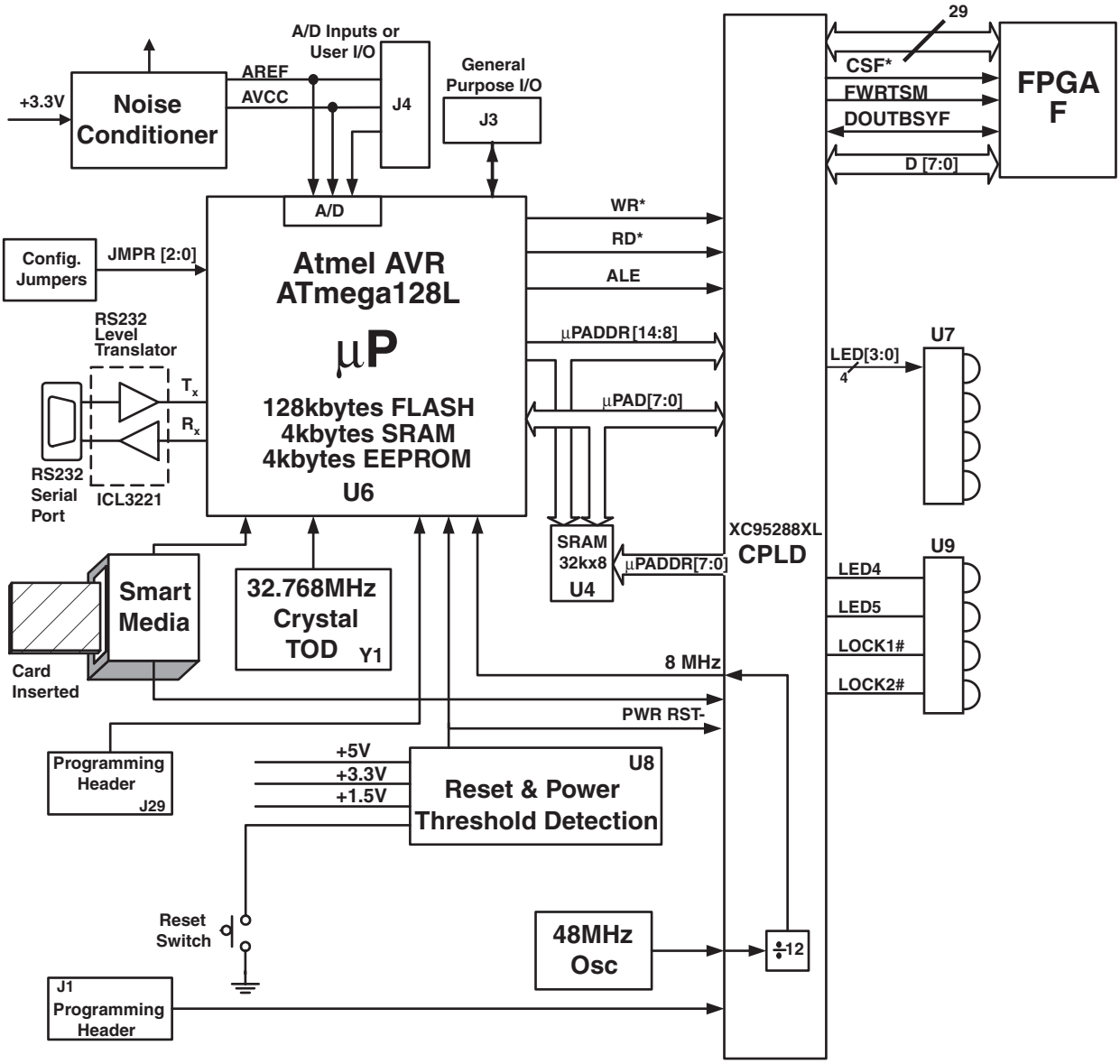


Figure 2-6 Block Diagram of ATmega128L and DN3000k10S Interfaces

A/D: Analog to Digital Converter

J4 connects to the A/D inputs of the ATmega128L. Header pins for AVCC and AREF are also provided if you wish to use AVCC elsewhere, or want to provide a cleaner AREF to the µP. The odd pins of **J4** are grounded, making for a clean connection with an IDC cable. The eight A/D pins (`UPADC[7:0]`) may also be used as TTL I/O. `UPADC[7:4]` also may be used to connect to the JTAG port of the ATmega128L (See "ATmega128L JTAG Interface" on page 2–12.).

According to Atmel documentation, the following features apply to the A/D:

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity

- ±2 LSB Absolute Accuracy
- 8 Multiplexed Single Ended Input Channels
- 7 Differential Input Channels
- 2 Differential Input Channels with Optional Gain of 10x and 200x
- Optional Left Adjustment for ADC Result Readout
- 0–VCC ADC Input Voltage Range
- Selectable Internal 2.56 V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceller

The **J4** Connections are shown in Figure 2-7. The AVCC port connections are shown in Figure 2-8.

+3.3 V is filtered via a 100µH (**L1**) inductor, a 100µF capacitor (**C47**), and a 0.1µF capacitor (**C46**) as shown in Figure 2-8. The output is AVCC (analog VCC). Inductor L1 is rated 120 ma. VREF is the result of a resistor division or **R16** and **R15**. The DN3000k10S is shipped with 30 ohm, 1% resistors (size 1210) in these locations, resulting in an AREF voltage of AVCC/2 or approximately 1.65 V. If you wish to supply your own AVCC and VREF voltages, remove **L1**, **R15**, and **R16**, and input your signals on **J4** pins 18 and 20. Remember that the ATmega128 also has an internal reference voltage of 2.56 V. If you use the internal 2.56 V reference, leave **C45** stuffed. Atmel documents state that externally decoupling AREF will improve the noise performance of the A/D.

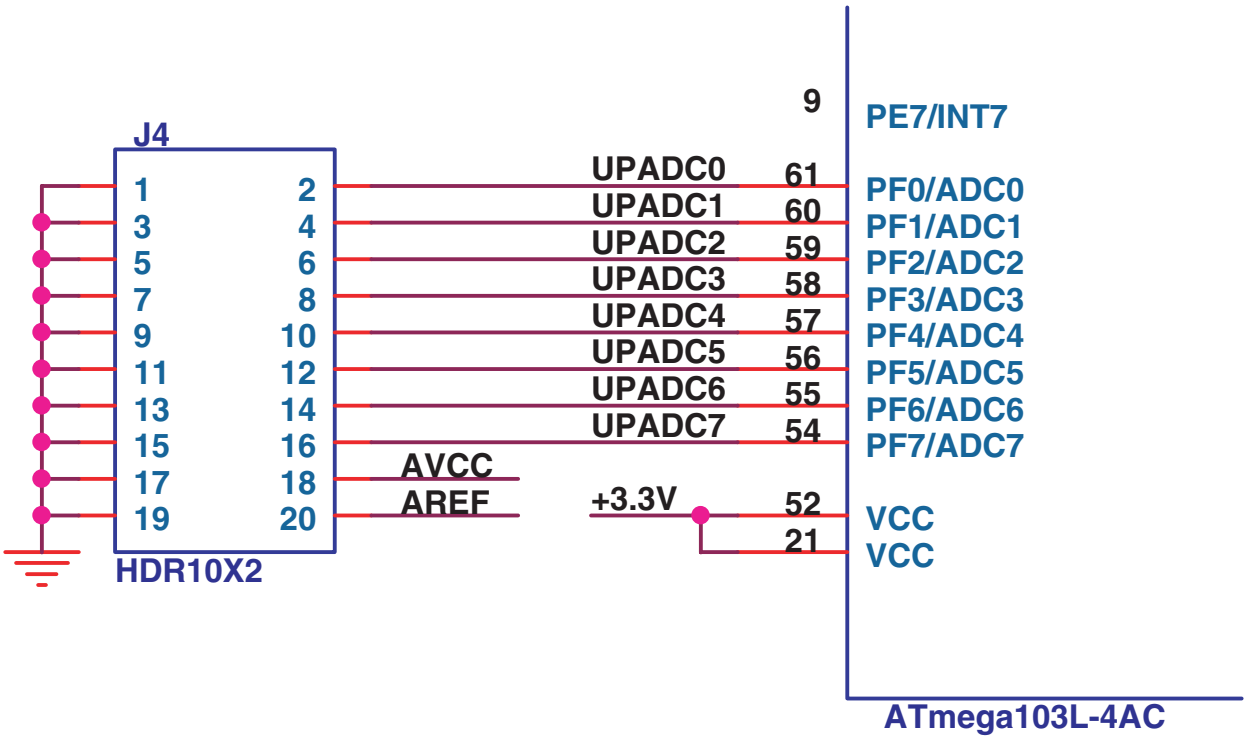


Figure 2-7 J4 Analog to Digital Connections

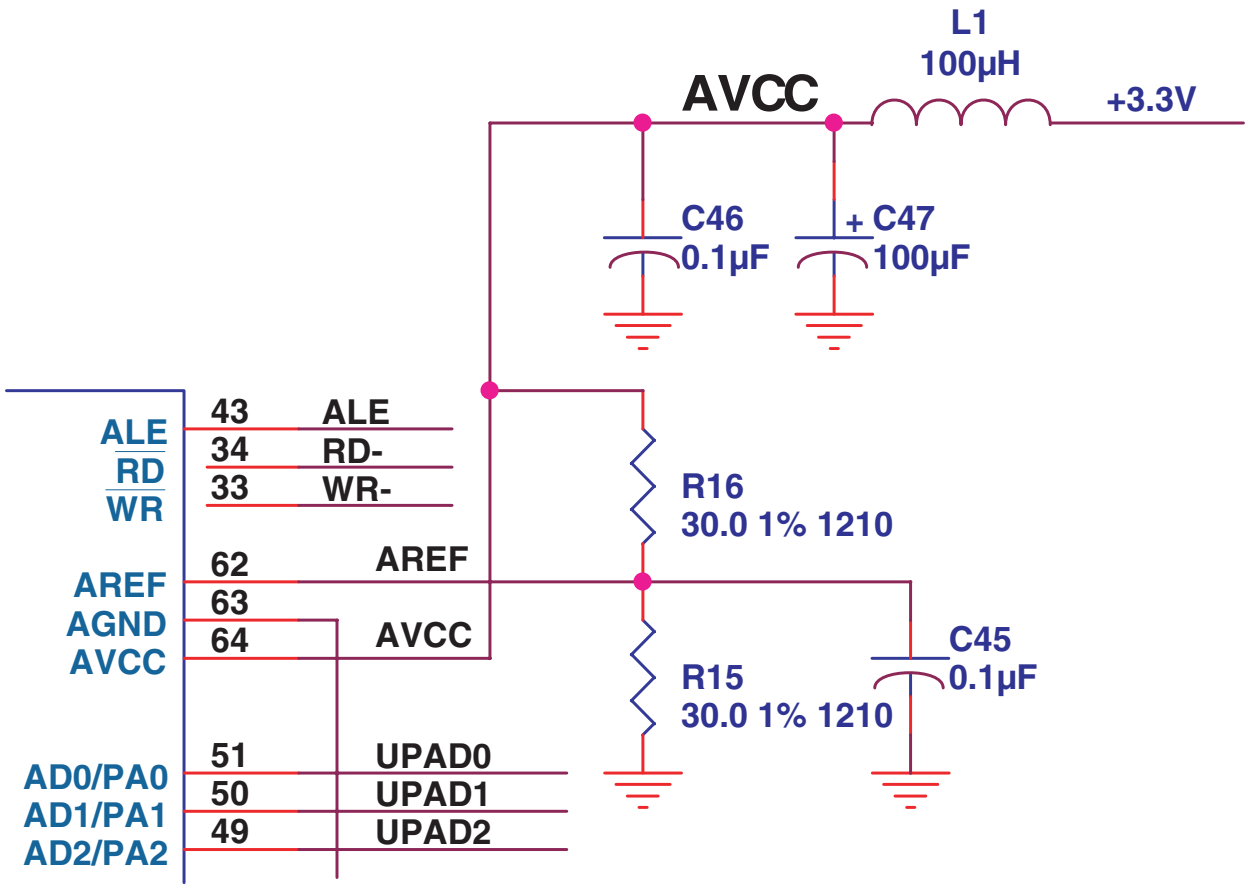


Figure 2-8 AVCC Connections

J3: Unused µP Connections

J3 contains connections to the ATmega128L that were not used elsewhere. These ten connections can be used for external TTL connections to the µP, externally generated interrupts, or any other function that the ATmega128L supports on these pins. Remember that the ATmega128L is not +5 V tolerant, so if you attach external TTL signals to these pins, the voltage level of these signals must not exceed +3.3 V.

The **J3** schematic is shown in Figure 2-9.

ATmega128L JTAG Interface

The ATMega128L processor has a JTAG interface that can be used for on-chip debugging, real-time emulation, and programming of FLASH, EEPROM, fuses, and Lock Bits. In order to take advangtage of the JTAG interface, you must hav ethe Atmel AVR JTAG ICE kit (part number ATAVR-JTAGICE) and AVR studio software that Atmel provides free at www.atmel.com. The JTAG interface for the ATmega128L can be accessed through four pins (**TCK**, **TMS**, **TDO** and **TDI**) on header **J4** of the DN3000k10S (see Figure 2-10).

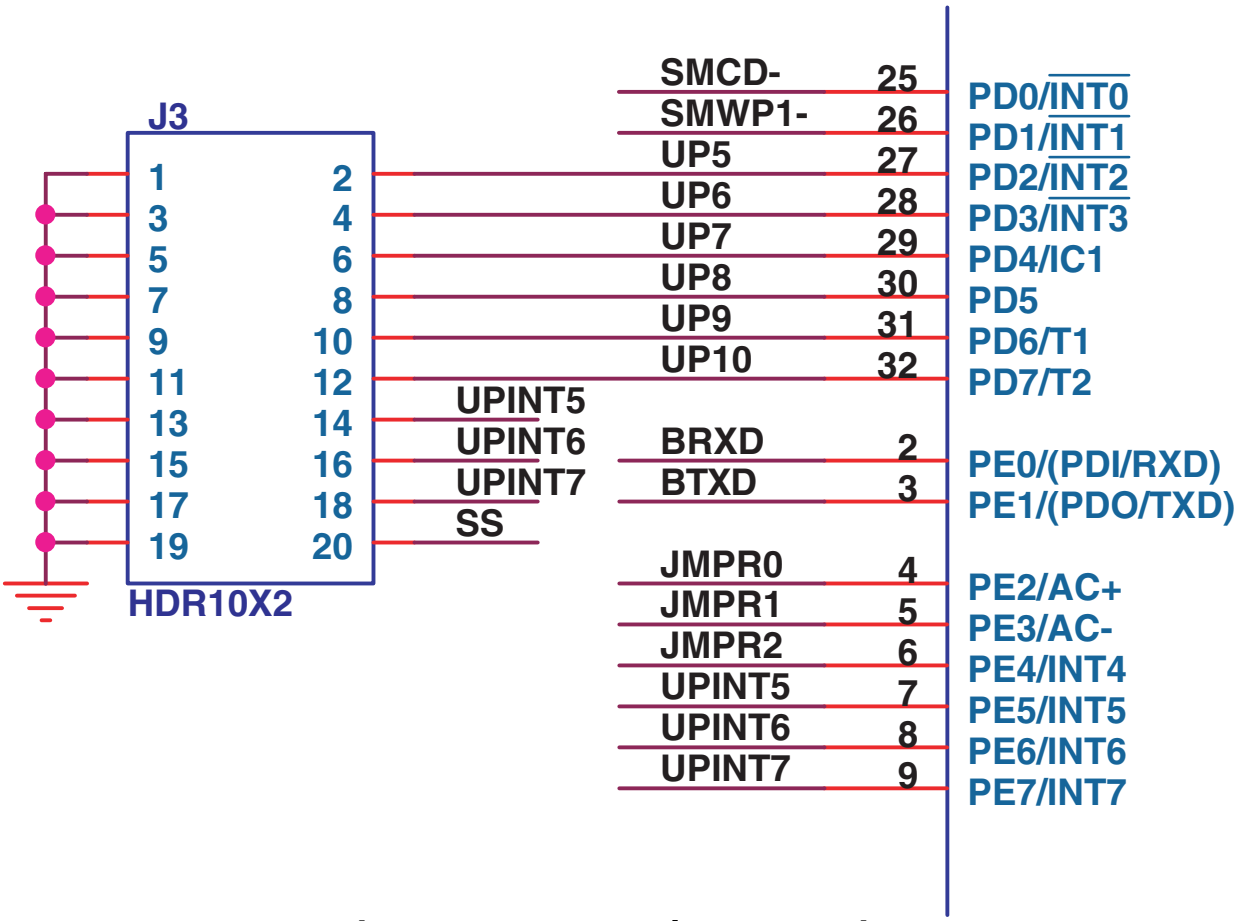


Figure 2-9 J3: Unused µP Connections

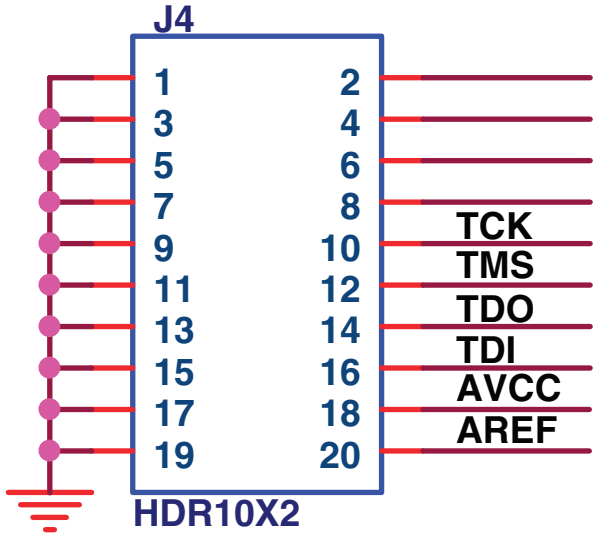


Figure 2-10 ATmega128L JTAG Interface

Programming the ATmega128L (U6)

A cable used to reprogram the ATmega128L is shipped with the DN3000k10S. You will need to reprogram the ATmega128L if we update the code or you intend to use the processor for your own application. **J29** is used for this purpose.

Figure 2-11 illustrates J29.

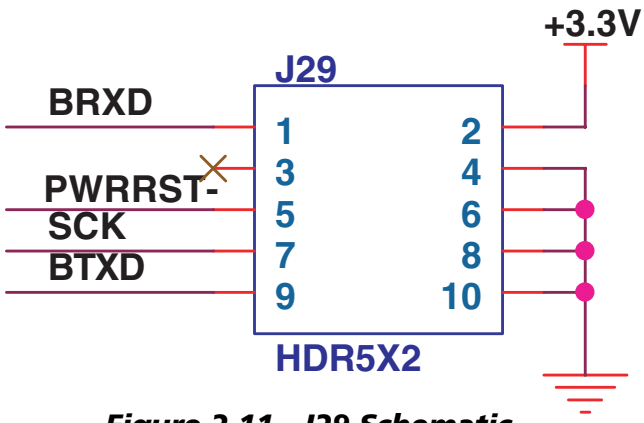


Figure 2-11 J29 Schematic

Detailed Instructions

1. Download the latest update for the processor and CPLD at **www.dini-group.com** (file `uP_CPLD.zip`).
2. You will first need to reprogram the CPLD. Please see "CPLD—XC95288XL" on page 2-16 for instructions (use the file `DNk10S_CPLD.jed` that can be found in the downloaded zip file.
3. Next, you will program the processor (ATmetga128L). Connect the AVR cable that was shipped with the DN3000k10S to header **J29** with the red/purple wire on the cable connected to pin 1 and connect the other end to the serial port of your PC.
4. In order to program the processor, you will need to install AVR Studio that is included on the Atmel CD that was shipped with the DN3000k10S. This software can also be downloadted at **www.atmel.com**.
5. From the Windows **START** menu, choose **PROGRAMS–>Atmel AVR Studio x.xx** (where x.xx is the version number).
6. Once AVR Studio is open, select **TOOLS–>STK500/AVRISP/JTAG ICE** and a new window should appear with the title **STK500**. At the bottom of the **STK500** window, if you see:

   ```
   Detecting…FAILED!
   ```

   that means either there is no power on the DN3000k10S, there is another program open that is using the serial port, or the serial cable connecting the AVR tool is not connected properly. If this happens, you should close down the window titled **STK500**, correct the situation, and then select **TOOLS–>STK500/AVRISP/JTAG ICE** again. You will not be able to continue unless you see something very similar to the following at the bottom fo the **STK500** window:

```
Detecting…AVRISP found on COM1:
Getting revisions…HW: 0x01, SW Major: 0x01, SW
Minor: 0x07…OK
```

7. On the **PROGRAM** tab, select the **ATmega128** under the **DEVICE** drop down menu, and in the **FLASH** section where it says **INPUT HEX FILE**, browse and select the file `DN3000k10S_128.a90` that can be found in the downloaded zip file (`uP_CPLD.zip`) from the Dini Group website. To program the device all you need to do is hit the **PROGRAM** button in the **FLASH** section. When the programming is complete (it takes about 45 seconds) you should see a message at the bottom of the window that looks something like this:

```
Detecting…AVRISP found on COM1:
Getting revisions…HW: 0x01, SW Major 0x01, SW
Minor: 0x07…OK
Reading FLASH input file…OK
Setting device parameters, serial programming
mode…OK
Entering programming mode…OK
Erasing device…OK
Programming FLASH using block mode…100% OK
Leaving programming mode…OK
```

8. After programming the processor, close all AVR Studio windows and setup the serial port according to the section titled "Setting up the Serial Port (J27 — RS232 Port)" on page 2-21. Please note that in this situation, connecting the serial port is mandatory and the FPGA cannot be configured via the SmartMedia card until you have completed all the instructions in this section.

9. Reset the DN3000k10S by pressing S1. After about 10 seconds, you should see the following in the HyperTerminal window:

```
**********NEED FPGA STUFFING INFORMATION*********

Enter number of FPGAs on Board (1—6):
```

Using the keyboard, enter the number of FPGAs on the board (should be 1 for the DN3000k10S). After you have entered this, you should see the following menu:

```
1) Virtex II 1000 (FG456)
2) Virtex II 6000 (FF1152)
3) Virtex II 4000 (FF1152)
4) Virtex II 3000 (FG676)

Please enter selection (1—4) for FPGA F:
```

Enter the type of FPGA that is stuffed on your DN3000k10S. If you enter the wrong type of FPGA or the incorrect number of FPGAs on the board, then you will need to reprogram the processor and follow these steps again.

10. The processor and the CPLD are now ready to configure the FPGA(s). Please see the section titled "Starting SelectMAP Configuration" on page 2-25 for further instructions.

CPLD—XC95288XL

Some non-volatile logic is needed to handle the counters and state machines associated with the high-speed interface to the SmartMedia card. We used an XC95288XL CPLD from Xilinx for this function. The datasheet is on the CD-ROM and is titled xc95288xl.pdf. Approximately 90% of the resources of this device are utilized, so 10% are available for your own purposes. The Verilog source for the CPLD is provided on the CD–ROM. The file name is CPLD.V.

The CPLD performs the following functions:

- Level Translation and logic inversion for LED[5:0], LOCK_N[2:1]
- µP SRAM Interface:
  - µP upper/lower address latch for 32K x 8 SRAM (**U4**): UPADDR[14:0]
  - SRAMCS—
- Interface to ATmega128L µP
  - Data Bus: UPAD[7:0]
  - Control: ALE, RD—, WR—
  - Clock: CPUCLK
- Interface to FPGA configuration signals
  - PROG—, DOUTBSYF, INITF—, DONEF
  - M[2:0]
  - SelectMap Interface
    + Data Bus: {D[7:1], DIND0F}
    + Chip Select: CSF—
    + Read/Write: FWRTSM—
  - JTAG: FTMS, FTDO, FTDI, FTCK
- Interface to SmartMedia Card
  - Data Bus: SMRTMED[7:0]
  - Control: SMCLE, SMALE, SMWE—, SMWP—, SMCE—, SMRE—, RDYBUSY—
- Interface to Serial FPGA Configuration Cable:
  - C_DONE, C_DIN, C_PROG—, C_INIT—, C_CCLK
- 30 Spare Connections to the FPGA
  - P3N[91:90], P4N[29:3], P2NX6

We may periodically update the CPLD. The CPLD can be reprogrammed using the Xilinx JTAG cable supplied with the DN3000k10S. The connections are on the 90° header on the top left corner of the PWB labeled **J1**. The relevant signals and the connections to **J1** are listed in Table 2-1. Figure 2-12 shows the location of **J1**. With the exception of V_{CC} and GND,

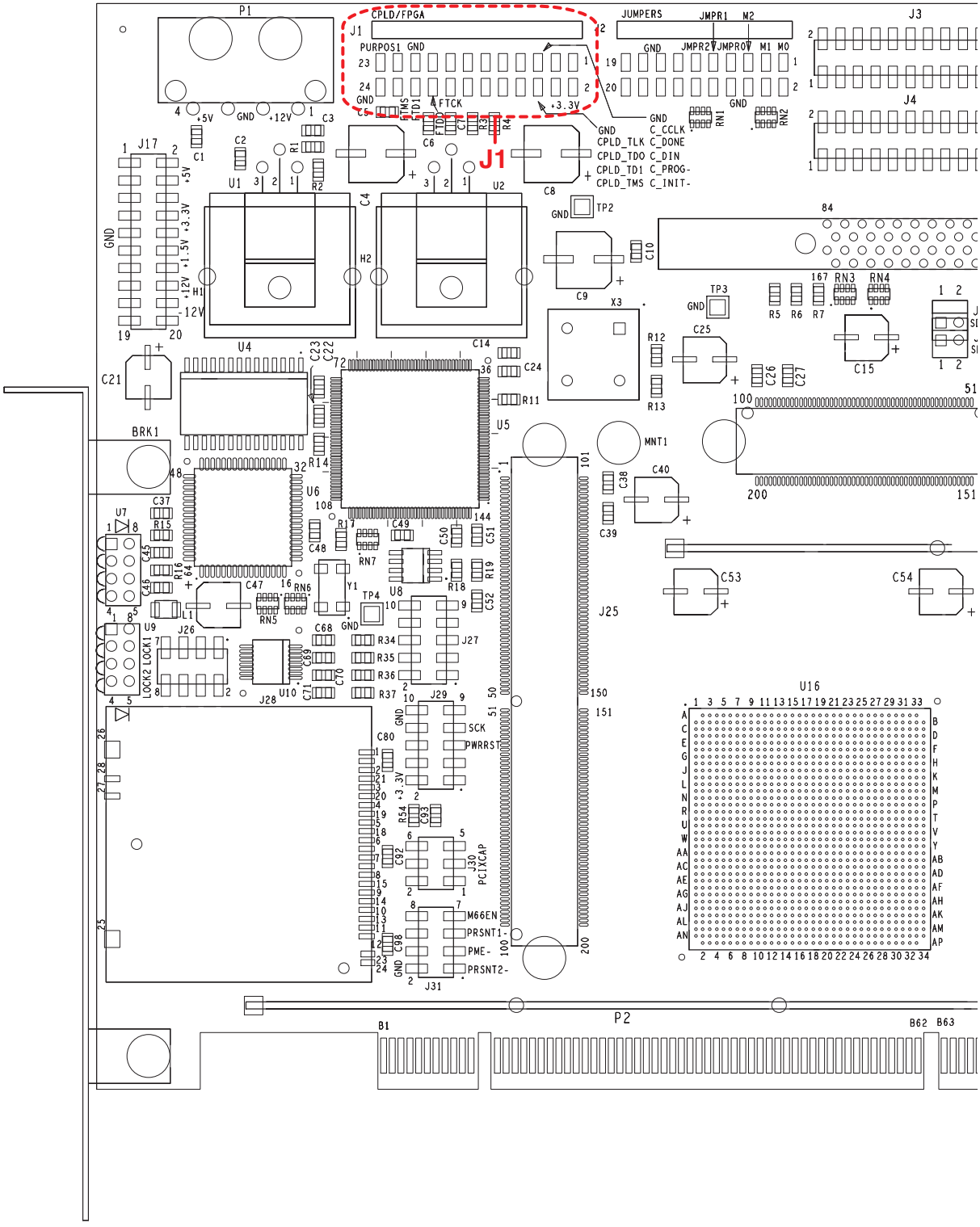


Figure 2-12 Location of J1 on the DN3000k10S

you will find that the signals are placed on the connector in the same order as the JTAG cable. A schematic of **J1** is shown in Figure 2-13.

Table 2-1 Signals and Connections to J1

JTAG Cable	Color	J1 Signal Name	J1 Pin
VCC	Red	+3.3 V	1, 2, 3, or 4
GND	Black	GND	5, 6, 17, 19, 21 or 24
TCK	Yellow	CPLD_TCK	8
TDO	Purple	CPLD_TDO	10
TDI	White	CPLD_TDI	12
TMS	Green	CPLD_TMS	14

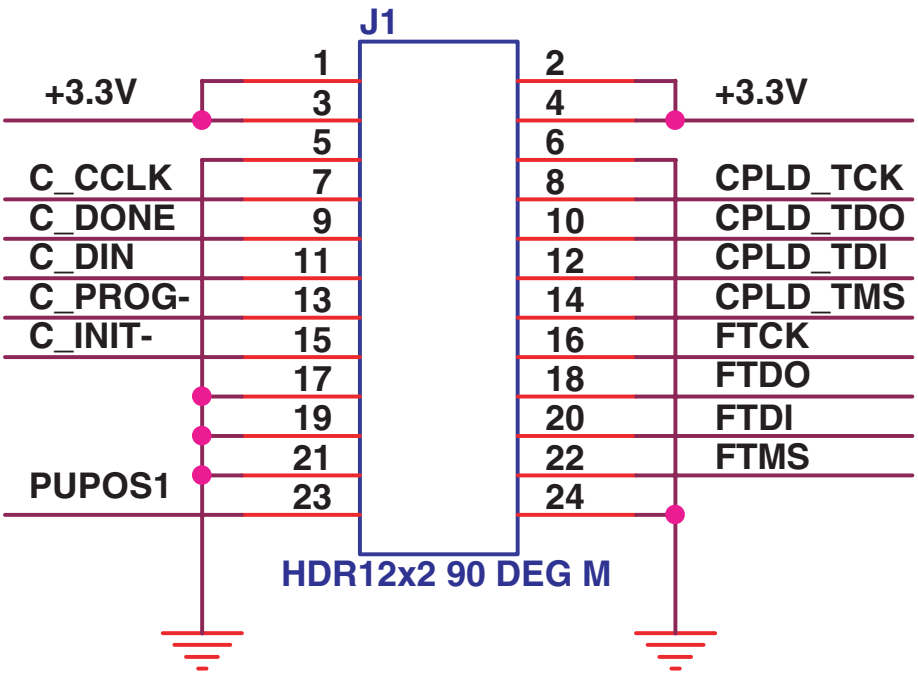


Figure 2-13 J1 CPLD JTAG Configuration, FPGA Serial Configuration and FPGA JTAG Configuration

Some Miscellaneous Notes on the CPLD

X3 is a 48 MHz oscillator. This part is soldered down to the PWB and is not intended to be user-configurable. The 48 MHz is divided down to 8 MHz in the CPLD to provide the clock for the ATmega128L µP. The processor clock signal is labeled CPUCLK (and BCPUCLK) on the schematic, and may have a note that describes the frequency as 4 MHz. Initially, we used an ATmega103L instead of the ATmega128L. The ATmega103L maximum frequency was 4 MHz.

The 48 MHz is used directly for the state machines in the CPLD for controlling the interface to the SmartMedia card. The frequency of 48 MHz is interesting because it is the closest frequency to 50 MHz that can be divided by an integer to get 8 MHz. The frequency 50 MHz is the fastest that the Xilinx VirtexII parts can be configured with SelectMap without wait states. So FPGA configuration using SelectMap occurs at very nearly the fastest theoretical speed.

Serial and JTAG configuration of the VirtexII FPGA are back off positions only—that is why those signals are connected to the CPLD. Xilinx has a long history of botching the configuration method in new FPGA families, so we made sure we had all possible options available.

If you want to use 100% of the CPLD and µP for your own purposes, you can configure the FPGA using the JTAG cable.

The 48 MHz clock can be divided down in the CPLD and used to drive the PWB clock network. See Chapter 4 for a more detailed description of this option.

The signals P3N[91:90], P4N[29:3], and P2NX6 are spare connections between the CPLD and the FPGA.

INITF– and DOUTBSY– are used by the CPLD and the µP for self-test purposes when the FPGA is configured with our reference design.

LOCK_N[2:1] are the logical invert of LOCK[2:1]. The lock LEDs should go **on** when the respective RoboclockII PLLs are locked.

Notes on Header J1

SelectMap using the SmartMedia card is the best way to configure the FPGA. Two other options exists if, for some reason, the SmartMedia media method is not working.

1. Serial Programming Using the Cable.
   Header J1 has the 5 serial connections that are used to configure the FPGA using the serial method. Table 2-4 has the pinouts. Note that this is a back-off position to SmartMedia and JTAG and should only be used in dire circumstances. Note also that header J1 will need to change to reflect “slave-serial” configuration.

Table 2-2 FPGA Serial Configuration Header

Name on Schematic:	Name on Cable	Cable Color	Header Pin
C_CCLK	CCLK	yellow	J1.7
C_DONE	D/P	blue	J1.9
C_DIN	DIN	green	J1.11
C_PROG–	PROG	orange	J1.13
C_INIT–	(none)	(none)	J1.15

Table 2-2 FPGA Serial Configuration Header

Name on Schematic:	Name on Cable	Cable Color	Header Pin
GND	GND	black	J1.5, J1.17, J1.19, J1.21, J1.6, J1.24
VCC	+3.3V	red	J1.1, J1.2, J1.3, J1.4

2. JTAG Programming.
   The JTAG connection can be used to configure the FPGA and can also be used to connect the ChipScope ILA Logic Analyzer (**www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=chipscope_ila**) or other solutions such as the Bridges2silicon system (see **www.bridges2silicon.com**). The JTAG method of configuration should be used if the SmartMedia method isn't working. Remember that a different bit file is necessary — the bit file must use the JTAG clock for configuration. Table 2-3 has the pinouts.

Table 2-3 FPGA JTAG Configuration Header

Name on Schematic	Name on Cable	Cable Color	Header Pin
FTCK	TCK	yellow	J1.16
FTDO	TDO	purple	J1.18
FTDI	TDI	white	J1.20
FTMS	TMS	green	J1.22
GND	GND	black	J1.5, J1.17, J1.19, J1.21, J1.6, J1.24
VCC	+3.3V	red	J1.1, J1.2, J1.3, J1.4

3. The signal `PUPOS1` on header **J1.23** serves no purpose.

SelectMAP Configuration Instructions

The FPGA on the DN3000K10S can be configured in SelectMAP mode using a Smart Media card. SelectMAP configuration is the easiest and quickest way to configure the FPGA. The DN3000k10S is shipped with two 32 MB Smart Media cards. One of these Smart Media cards contains a reference design bit file produced for SelectMAP configuration, and a file `main.txt` that sets options for the configuration process (for description of options, see "Creating Main Configuration File main.txt" on page 2-23). This Smart Media card has been marked as read-only by the silver, circular sticker on the card. The other Smart Media card is empty and is for use with your own designs. To configure the FPGA with the reference design, please skip to "Starting SelectMAP Configuration" on page 2-25.

Creating Bit Files for SelectMAP

Create bit files with Xilinx 3.3i with service pack 8 or Xilinx ISE 4.x

- Use Xilinx 3.3i with service pack 8 and the patch for VirtexII (see **http://support.xilinx.com/techdocs/11805.htm**) or Xilinx ISE 4.

For Xilinx 3.3i Design Manager:

- After creating a project in Xilinx, go to the menu **Design/Options** and choose **Edit Options for Configuration** and uncheck the box. Enable the **Power Down Status pin (Done pin)** on the **Configuration** tab. Also make sure that on the **Startup** tab, the **Startup Clock** is CCLK.

For Xilinx ISE 4.x Design Manager:

- After creating a project in Xilinx, go to the menu **Design/Options** and choose **Edit Options for Configuration** and go to the **Startup** tab and make sure the **Startup Clock** is CCLK; on the **Readback** tab select SelectMap.

For Xilinx ISE 4.x Project Navigator

- NOTE: All the bitgen options are set correctly by default; however, you may want to double-check a few of the options. After creating a project and adding your source file, right-click on **Generate Programming File** in the **Process View** window. Select the **Startup Options** tab and make sure that the **Startup Clock** is set to CCLK. Select the **Configuration Options** tab and make sure C**onfiguration Pin Powerdown** has a value of Pull Up.

Setting up the Serial Port (J27 — RS232 Port)

J27 is for an RS232 connection to a terminal. An ICL3221 (**U10**) provides voltage translation to RS232 levels. A cable that converts the 10-pin header to a DB9 is shipped with the DN3000k10S. This cable comes packaged with a bracket attached. Remove the bracket to eliminate the possibility of it falling on the DN3000k10S, which could short signals and damage the board. After you have removed the bracket, plug the cable into **J27**. **J27** is not keyed—so make sure you get the orientation correct. Pin 2 is clearly labeled. Pin 1 is opposite of Pin 2, and PIn 1 is identified with a dot. Figure 2-14 is a cutout from the assembly drawing, and shows the location of **J27** and Pin 1.

A female-to-female RS232 cable is provided with the DN3000k10S. This cable will attach directly to the RS232 port of a PC. We get our cables from Jameco (**http://www.jameco.com**).The part number is 132345. Male-to-female extension cables are part number 25700.

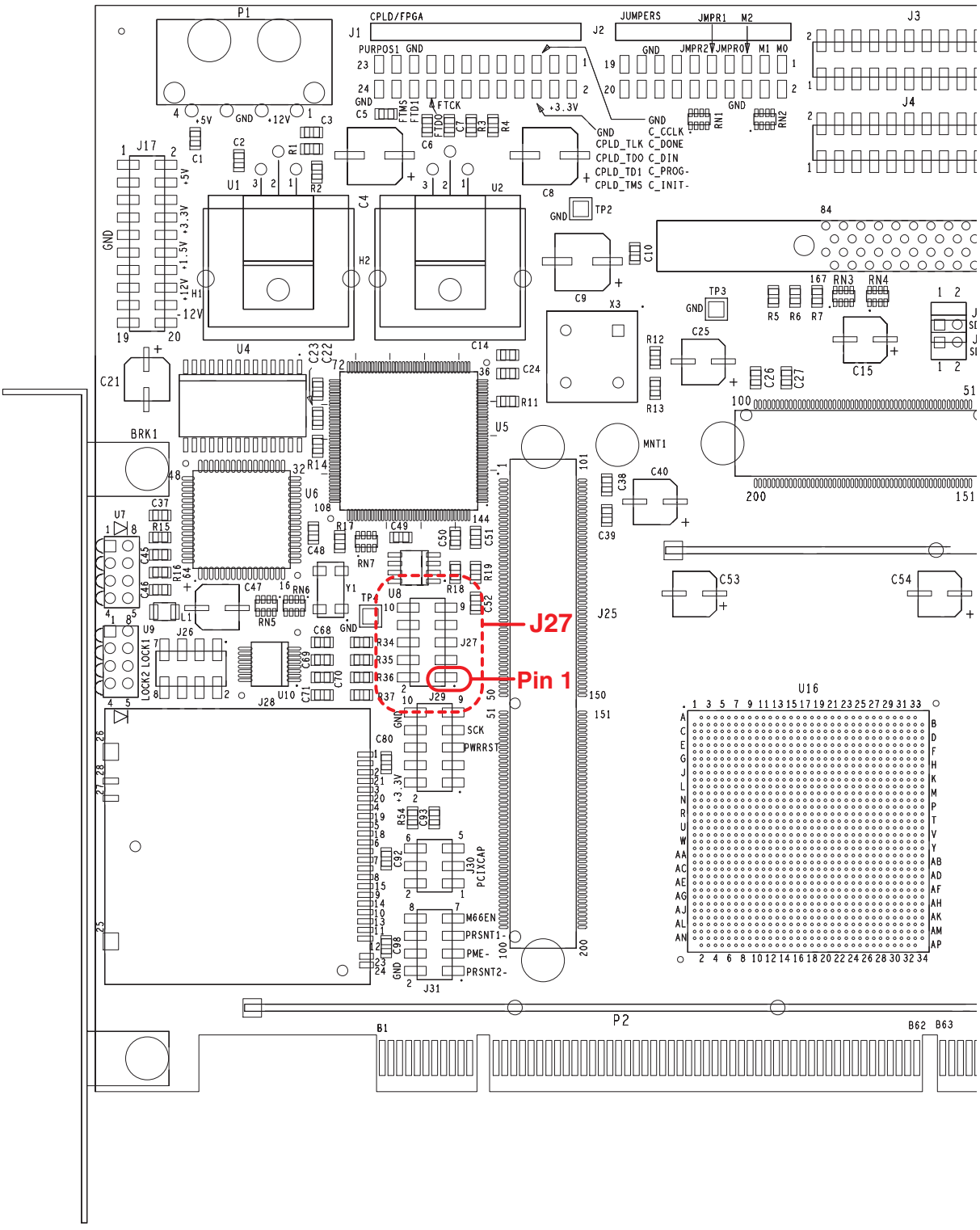


Figure 2-14 J27—RS232 Port Assembly Drawing

The RS232 port is configured with the following parameters:

Bits per second:	9600
Data bits:	8
Parity:	None
Stop Bits:	1
FLow control:	None
Terminal Emulation:	VT100

We use the Windows-based program HyperTerminal (`Hypertrm.exe`). The configuration file `DN3000k10S.ht` is supplied on the CD-ROM or can be downloaded from our web page.

Users have the option of connecting the serial port if they wish to see any messages during the configuration process.

NOTE: It is NOT mandatory to have the serial port connection in order to configure the FPGA in SelectMAP mode. However, if an error occurs during the configuration, then without a serial port connection the user will not be able to see any error messages. In addition, without a serial port connection, a user cannot select any Main Menu options after the configuration process is complete.

Creating Main Configuration File main.txt

To control which bit file on the Smart Media card is used to configure the FPGA in SelectMAP mode a file named `main.txt` must be created and copied to the root directory of the Smart Media card. The configuration process cannot be performed without this file. Below is a description of the options that can be set in the file, a description of the format this file needs to follow, and an example of a `main.txt` file.

Options:

Verbose Level — During the configuration process, there are three different verbose levels that can be selected for the serial port messages:

- Level 0:
  - Fatal error messages
  - Bit file errors (e.g., bit file was created for the wrong part, bit file was created with wrong version of Xilinx tools, or bitgen options are set incorrectly)
  - Initializing message will appear before configuration
  - A single message will appear once the FPGA is configured
- Level 1:
  - All messages that Level 0 displays
  - Displays configuration type (should be SelectMAP)
  - Displays current FPGA being configured if the configuration type is set to SelectMAP
  - Displays a message at the completion of configuration for each FPGA configured.
- Level 2:
  - All messages that Level 1 displays
  - Options that are found in `main.txt`
  - Bit file names for each FPGA as entered in `main.txt`

- Maker ID, device ID, and size of Smart Media card
- All files found on Smart Media card
- If sanity check is chosen, the bit file attributes will be displayed (part, package, date, and time of the bit file)
- During configuration, a "." will be printed out after each block (16 KB) has successfully been transferred from the Smart Media to the current FPGA.

Sanity Check — The Sanity Check if enabled, verifies that the bit file was created for the right part, the right version of Xilinx was used, and the bitgen options were set correctly. If any of the settings found in the bit file are not compatible with the FPGA, a message will appear from the serial port, and the user will be asked whether or not they want to continue with the bit file. Please see the section "Creating Bit Files for SelectMAP" on page 2-21 for details on which bitgen options need to be changed from the default settings. A PC version of the sanity check can be run on your bit files before copying them onto the Smart Media card; see section "PC Bit File Sanity Check" on page 2-27 for more details.

Format:

The format of the `main.txt` file is as follows:

- The first nonempty/uncommented line in `main.txt` should be:

  ```
  Verbose level: X
  ```

  where "X" can be 0, 1 or 2. If this line is missing or X is an invalid level, then the default verbose level will be 2.

- The second nonempty/uncommented line in `main.txt` tells whether or not to perform a sanity check on the bit files before configuring an FPGA:

  ```
  Sanity check: y
  ```

  where "y" stands for yes, "n" for no. If the line is missing or the character after the ":" is not "y" or "n" then the sanity check will be enabled.

- For each FPGA that the user wants to configure, there should be exactly one entry in the `main.txt` file with the following format:

  ```
  FPGA F: example.bit
  ```

  In the above format, the "F" following FPGA is to signal that this entry is for FPGA F, and FPGA F would then be configured with the bit file `example.bit`. The DN3000k10S only has one FPGA, which is FPGA F. There can be any number of spaces between the ":" and the configuration file name, but they need to be on the same line.

- Comments are allowed with the following rules:
  1. All comments must start at the beginning of the line.
  2. All comments must begin with //

3. If a comment spans multiple lines, then each line should start with //

   Commented lines will be ignored during configuration, and are only for the user's purpose.

- The file `main.txt` is **NOT** case sensitive.

IMPORTANT: All configuration file names have a maximum length of eight (8) characters, with an additional three (3) for the extension. Do not name your configuration bit files with long file names. In addition, all file names should be located in the root directory of the Smart Media card—no subdirectories or folders are allowed. Since the `main.txt` file controls which bit file is used to configure the FPGA, the Smart Media card can contain other bit files.

Example of `main.txt`:

```
//start of file "main.txt"
Verbose level: 2
Sanity check: y

FPGA F: fpgaF.bit
//the line above configures FPGA F with the bit file
"fpgaF.bit"

//end of main.txt
```

Given the above example file:

- Verbose level is set to 2
- A sanity check on the bit files will be performed
- FPGA F will be configured with file `fpgaF.bit`.

Starting SelectMAP Configuration

If using the reference design SmartMedia card that came with the DN3000k10S then no files need to be copied to the card. Otherwise, copy your bit file and `main.txt` to the root directory of the SmartMedia card using the FlashPath floppy adapter. Make sure the jumpers on **J2** are set for SelectMAP as shown in Table 2-4.

Table 2-4 J2 Configuration Jumper Settings

Pins 1 & 2	Pins 3 & 4	Pins 5 & 6	Configuration Mode
No Jumper	Jumper	No Jumper	JTAG
Jumper	No Jumper	No Jumper	SelectMAP

Set up the serial port connection as described above in "Setting up the Serial Port (J27 — RS232 Port)" on page 2-21. Next, place the SmartMedia card in the SmartMedia socket on the DN3000k10S and turn on the power (NOTE: the card can only go in one way). The SmartMedia card is hot-swappable and can be taken out or put into the socket even when the power is on. Once the power has been turned on, the configuration process will begin as long as there is a valid SmartMedia card inserted

properly in the socket. If there is not a valid SmartMedia card in the socket, then **LED1** will be lit (see Figure 8-2 on page 8-3 for LED descriptions) and the Main Menu will appear from the serail port. A SmartMedia card is detemined to be invalid if either the format of the card does not follow the SSFDC specifications, or if it does not contain a file named `main.txt` in the root directory. If the configuration was successful, a message stating so will appear and the **Main Menu** will come up. Otherwise, an error message will appear.

The LEDs on **U7** and **U9** give feedback during and after the configuration process; see "LEDs" on page 8-3 for further details.

After the FPGA has been configured, the following **Main Menu** will appear on the serial port:

```
1. Configure FPGA(s) using main.txt
2. Interactive FPGA configuration menu
3. Check Configuration status
4. Select file to use in place of main.txt
5. List files on SmartMedia
6. Memory test (BlockRAM, SSRAM, SDRAM)
```

Description of Main Menu Options.

1. **Configure FPGAS in Using "`main.txt`" as the Configuration File**— By selecting this option, the FPGA will configure in SelectMAP mode. You can also press the reset button (**S1**) to reconfigure the FPGA in SelectMAP mode.
2. **Interactive FPGA configuration menu** — This option takes you to a menu titled "**Interactive Configuration Menu**" and allows the FPGA to be configured through a set of menu options instead of using the `main.txt` file. The menu options are described below.

   Description of **Interactive Configuration Menu** options:

   1. **Select a bit file to configure FPGA(s)** — This menu option allows the user to select a bit file from a list of bit files found on the SmartMedia card to use to configure the FPGA.
   2. **Set verbose level (current level = 2)** — This menu option allows the user to change the verbose level from the current setting. Please note: if the user goes back to the main menu and configures the FPGA(s) using `main.txt`, the verbose level will be set to whatever setting is specified in `main.txt`.
   3. **Disable/Enable sanity check for bit files** — This menu option either allows the user to disable or enable the sanity check, depending on what the current setting is. Please note: if the user goes back to the main menu and configurest the FPGA(s) using `main.txt`, the sanity check will be set to whatever setting is specified in `main.txt`.

M) **Main menu** — This menu option takes the user back to the Main Menu described above.

3. **Check Configuration status** — This option checks the staus of the **DONE** pin and prints out whether or not the FPGA(s) have been configured along with the file name that was used for configuration.
4. **Select file to use in place of** `main.txt` — By default, the processor uses the file `main.txt` to get the name of the bit file to be used for configuration as well as options for the configuration process. However, a user can put several files that follow the format for `main.txt` on the SmartMedia card that contain different options for the configuration process. By selecting the main menu option 4, the user can select a bit file from a list of files that should be used in place of `main.txt`. After selecting a new file to use in place of `main.txt`, the user should select Main Menu option 1 to configure the FPGA(s) according to this new file. If the power is turned off or the reset button (**S1**) is pressed, the configuration file is changed back to the default, `main.txt`.
5. **List files on SmartMedia** — This option prints out al ist of all the files found on the SmartMedia card.
6. **Memory test (BlockRAM, SSRAM, SDRAM)** — This option will only appear if the FPGA is configured. In addition, the test will only be run if the FPGA is configured with the reference bit file that was shipped with the DN3000k10S.

PC Bit File Sanity Check

A version of the sanity check has been compiled for use on a PC; the executable is `sanityCheck.exe`, which can be found on the CD shipped with the DN3000k10S. This allows you to run the sanity check on bit files before copying them onto the Smart Media card. This PC bit file sanity check verifies that the right version of Xilinx tools was used and the bitgen options have been set correctly. To run the sanity check from the command line:

```
%sanityCheck –f fpga.bit –d –s
```

Command line options:

- The -f option is required and must be followed by the name of the bit file to perform the sanity check on.
- The -d option is optional. It prints out a description of the different bitgen options and their different values.
- The -s option is optional. It prints out the current bitgen settings found in the file specified with the -f option.

Expected output:

- If the bit file passes the sanity check, you should see something similar to:

```
$ sanityCheck -f fpga_sm.bit

** Performing Sanity Check on File: fpga_sm.bit **
```

```
DATE: 2001/10/01
TIME: 10:47:01
PART: 2V6000ff1152
FILE SIZE = 2470068 bytes

ALL BITGEN OPTIONS ARE SET CORRECTLY
```

- If the bit file does not pass, then a message stating why it didn't pass will print out. For example:

```
$ sanityCheck -f fpga_sm.bit

** Performing Sanity Check on File: fpga_sm.bit **

DATE: 2001/10/01
TIME: 10:47:01
PART: 2v6000ff1152
FILE SIZE = 2470068 bytes
ERROR:  PowerDown status pin is enabled, you must
        disable this option to configure the FPGA in
        SelectMAP mode.
```

SmartMedia

The configuration bit file for the FPGA is copied to a SmartMedia card using the SmartDisk FlashPath Floppy Disk Adapter. The approximate file size for each possible VirtexII FPGA is shown below in Table 2-5. Note that several BIT files can be put on a 16-megabyte card. We supply two 16-megabyte SmartMedia cards with the DN3000k10S. SmartMedia is a standard, so you can get more SmartMedia cards if you want. The DN3000k10S requies a +3.3 V card. Card sizes of 16, 32, 64 and 128 megabytes have been tested on the DN3000k10S. We have not seen 256 MB or larger cards for sale yet, but when we do there will probably be an update to the CPLD and processor on our website to support them. .

Table 2-5 VirtexII FPGA Approximate File Sizes

VirtexII FPGA	Number of Configuration Bytes
XC2V3000	1,311,804
XC2V4000	1,957,500
XC2V6000	2,731,196
XC2V8000	3,632,892

We get our SmartMedia cards from **http://www.computers4sure.com/**. A Delkin Devices 16-megabyte card (part number DDSMFLS2-16) sells for about $15. A 32-megabyte card (part number DDSMFLS2-32) will set you

back about $20 (see Figure 2-15). New SmartMedia cards require formatting before use.

NOTE: SmartMedia cards must be formatted before they are used. The Windows format command does not work—it is necessary to use the FlashPath utility to format a SmartMedia card.

Figure 2-15 Delkin 32 MB 3.3 V Smart Media Card

Do not press down on the top of the SmartMedia Connector **J28** if a SmartMedia card is not installed. The metal case shorts to the +3.3 V power supply and the case gets hot enough to burn your finger. We suggest that you leave a SmartMedia card in the connector to prevent this from occurring.

NOTE: Do NOT press on the SmartMedia Connector J28 if a card is not installed!

WARNING:

Do NOT format a SmartMedia card using the default Windows format program. All Smart Media cards come preformatted from the factory, and files can be deleted from the card when they are no longer needed. If for some reason you absolutely need to format a SmartMedia card, you must use the format program that is included in the FlashPath (SmartMedia floppy adapter) software.

Synthesis and Emulation Issues

We use the following tools for synthesis:

Synplicity Symplify (http://www.synplicity.com/)

Synopsys FPGA Express (http://www.synopsys.com/)

Synopsys FPGA Compiler II

Exemplar LeonardoSpectrum
(http://www.exemplar.com/products/leonardospectrum.html)

Of the four listed here, we find that Synplicity offers the best performance, followed by Exemplar. The Synopsys products are not the easiest products to use, and probably should be avoided until Synopsys decides that they want to be in this market. It is generally not worth your time to preserve your Synopsys ASIC compiler directives and scripts by using the FPGA synthesis products from Synopsis. The time you save using Synopsys products is offset by other hassles.

Synthesis Notes

1. The FPGA used on your DN3000k10S is either a 2v4000, or 2v8000 in an FF1152 package. Unless you paid for a faster speed grade, the –4 is what you will be getting.
2. Memories are best implemented by describing them behaviorally in your RTL. All four synthesis products are sophisticated enough to map your behavioral descriptions into the memory blocks. It is **NOT** necessary to instantiate Xilinx memories manually. Make sure, however, to check the report files to make sure that your memories were implemented in memory blocks (if this is possible). Sometimes subtle changes are needed to your RTL to get the synthesis programs to recognize that you intended to use an embedded memory block.
3. Much to our surprise, the synthesis programs recognized RTL multiplier code and used the embedded multipliers without any trouble. So, like the memories, RTL description of your multipliers is all that is necessary. Make sure to check the report files—multipliers that are implemented using logic blocks (as opposed to the embedded memory blocks) take huge amounts of FPGA resources.
4. Clocks are the biggest problem when converting ASIC code to FPGA code. FPGAs only have a limited number of clock arrays. This is far too complicated to describe here, so get the *Virtex II User's Manual* and read about the clocks.

Chapter 3

PCI

Overview

The DN3000k10S can be hosted in a 32-bit or 64-bit PCI slot. PCI-X is also supported. Standalone operation is described in "Stand-Alone Operation" on page 6-4. A 2v6000-4, with care, should be able to support a 64-bit, 66 MHz PCI or PCI-X controller. We have not tested the PWB at PCI-X speeds of 100 MHz and 133 MHz. We suspect, but won't guarantee, that the DN3000k10S can support these high frequencies, provided the speed grade of the FPGA is adequate. Figure 3-1 shows the FPGA pin connections for the PCI signals. This data is provided on the CD-ROM in a UCF file titled `fpga.ucf`, for your convenience.

The PCI/PCI-X edge connector is shown in Figure 3-2.

VirtexII parts cannot tolerate +5 V TTL signaling, so the DN3000k10S must be plugged into a +3.3 V PCI slot. PCI-X, by definition, is +3.3 V signaling. The PWB is keyed so that it is not possible to mistakenly plug the board into a +5 V PCI slot. Do NOT grind out the key in the PCI host slot, and do NOT modify the DN3000k10S to get it to fit into the slot. If you need a +3.3 V PCI slot, the DNPCIEXT-S3 Extender card can do this function. The link is **http://www.dinigroup.com/products/pciextender.html**. This extender also has the capability to slow the clock frequency of the PCI bus by a factor of two—a function that is very useful when prototyping ASICs.

NOTE: +5 V Signaling on VirtexII parts causes them to smoke! This is quite BAD!

Do NOT modify the DN3000k10S board to fit into your PCI sxlot.

PCI Mechanical Specifications

The DN3000k10S is **not** a standard sized PCI card—it is too tall. The board is compliant to the PCI specification for the length, 12.25 inches long, but it is 5.25 inches high. This is sometimes an issue in servers that have a bracket installed over the top of the PCI cards. If you need to close the case on a DN3000k10S, most tower configurations should work. Figure 3-3 shows the exact dimensions of the DN3000k10S.

Some Notes on the DN3000K10S and PCI/PCI-X

+3.3 V power is not needed on the host PCI connector. +3.3 V power is derived from +5 V using an on-board 5 A liner regulator. Power distribution for the DN3000k10S is described in "Power Supplies and Power Distribution" on page 6-1.

`LOCK#` has a pull-up. This is technically a violation of the PCI specification, but we have seen systems (from SUN!) that have the `LOCK#` pin floating.

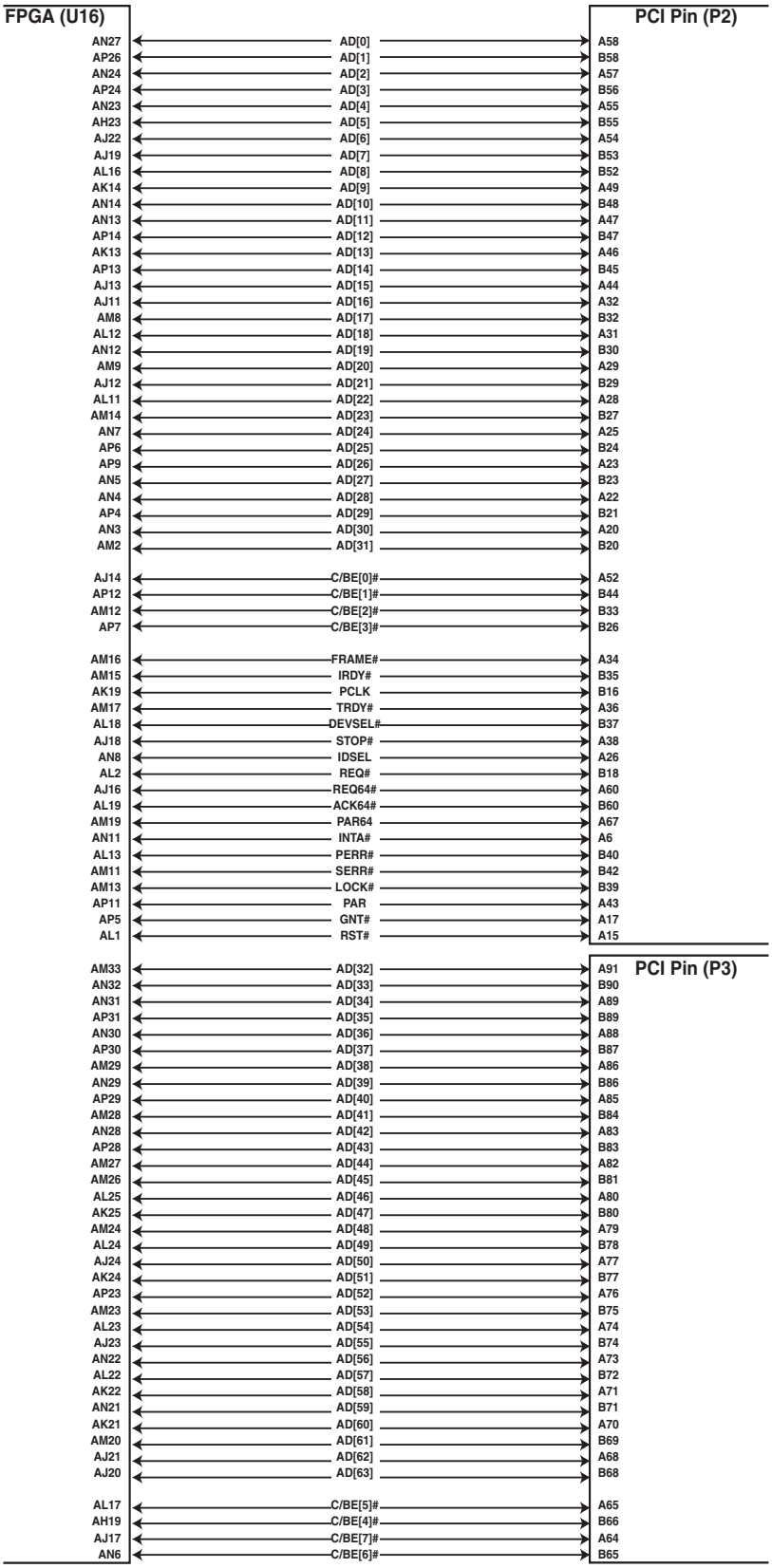


Figure 3-1 FPGA Pin Connections for PCI Signals

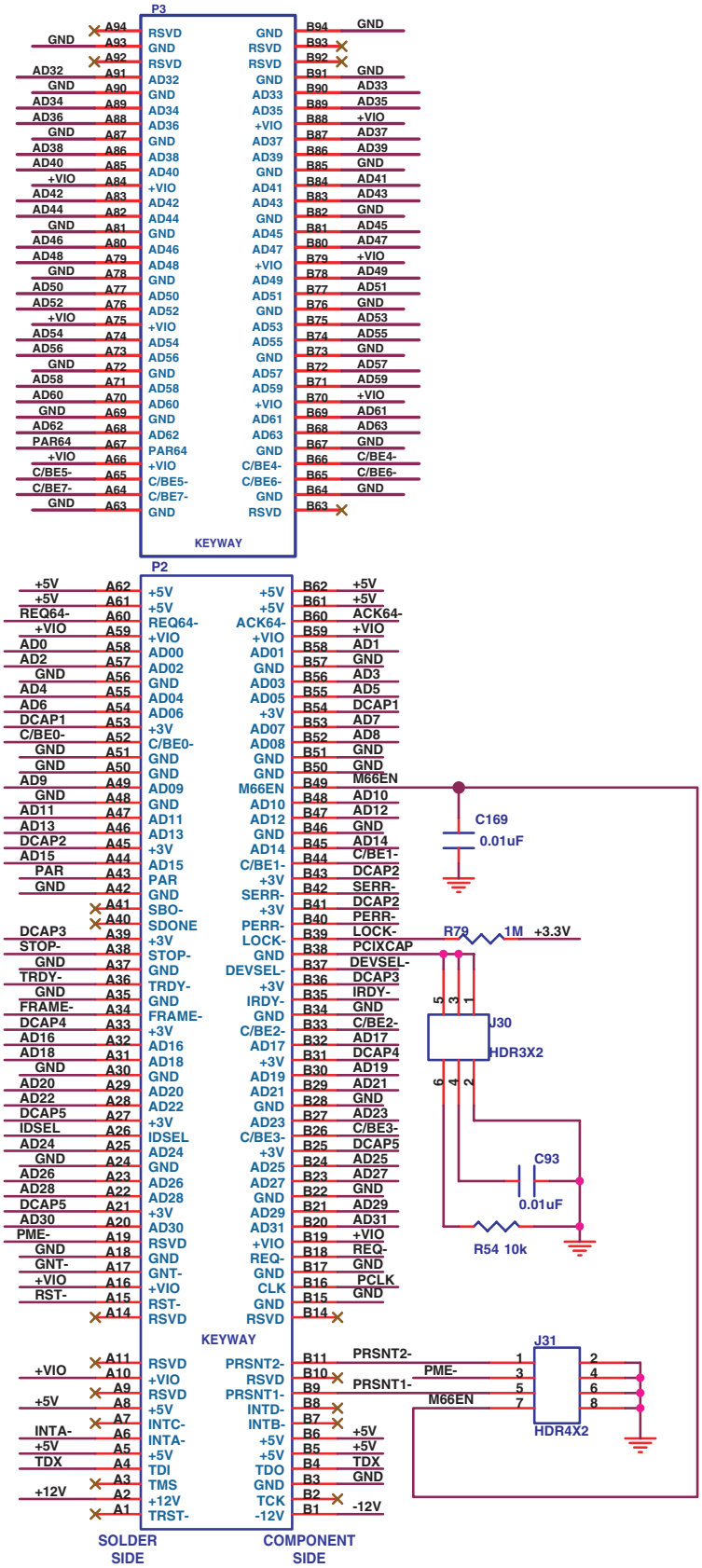


Figure 3-2 PCI/PCI-X Edge Connector

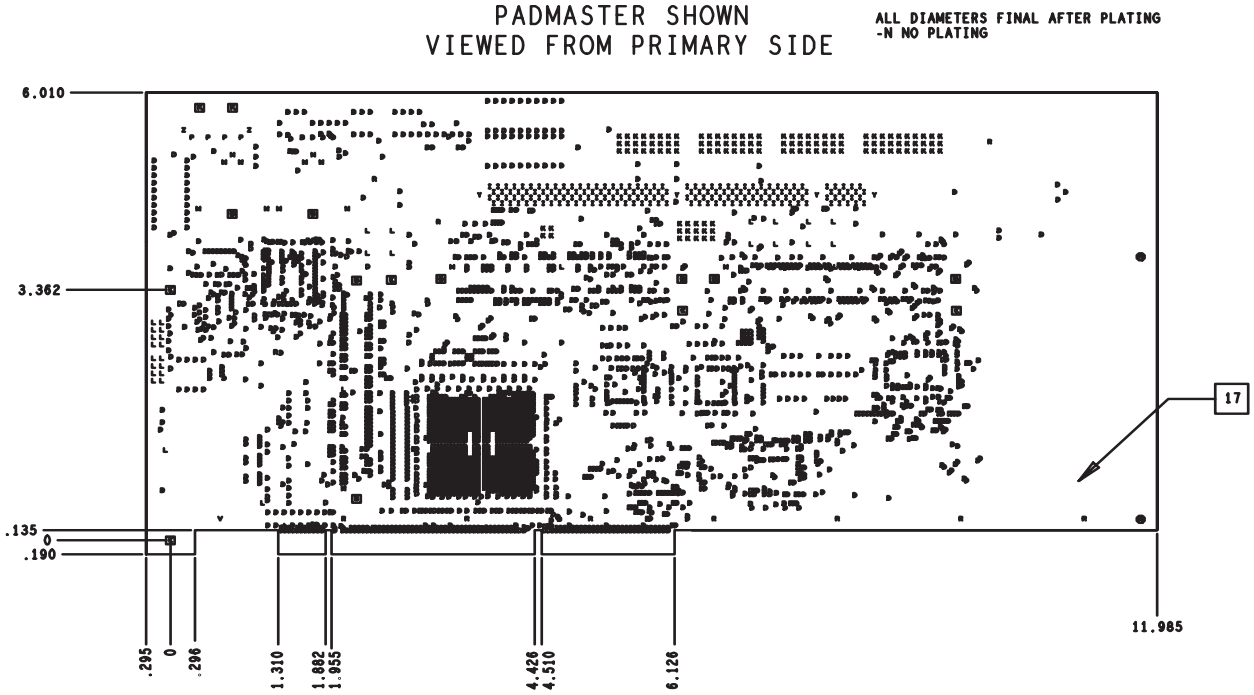


Figure 3-3 DN3000k10S Dimensions

Remember that the function of this pin was deleted in the 2.2 version of the PCI Specification. The pull-up is 1M, which should not adversely impact PCI functionality in any way.

The PCI JTAG signals TDI, TDO, TCK, TRST#, are not used. TDI and TDO are connected together per the PCI Specification to maintain JTAG chain integrity on the motherboard. The signals TMS, TCK, and TRST# are left unconnected.

The VirtexII FPGAs are not +5 V tolerant, so you must plug the DN3000k10S into a +3.3 V PCI slot. Do NOT modify the connector to get the board to fit. If you need +5 V to +3.3 V PCI voltage translation, get one of our extenders. The link is:

http://www.dinigroup.com/products/pciextender.html.

The FPGA is volatile, meaning it loses its brains when power is off. The SmartMedia method takes about 1 second to configure a 2V6000 after power is stable. It is likely that the FPGA will finish the configuration process before RST# is deasserted. If your system has an unusually fast RST#, it is possible that the FPGA will not be configured when RST# deasserts. A RST# that deasserts before the FPGA has finished cannot properly configure the PCI/PCI-X mode latch.

The signal 3.3Vaux is not connected.

The signals INTB#, INTC#, and INTD# are not connected.

J31: Present Signals for PCI/PCI-X (Pins 1–2 and 5–6)

The present signals indicate to the system board whether an add-in card is physically present in the slot and, if one is present, the total power requirements of the add-in card.

The **J31** PCI-X Present Header is shown in Figure 3-4.

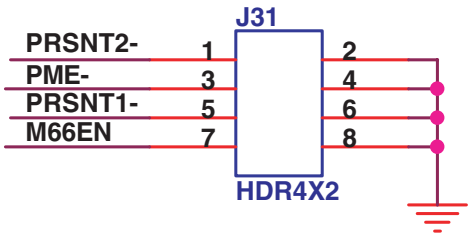


Figure 3-4 J31 PCI-X Present Header

Table 3-1 shows the Present Signal Definitions for PCI/PCI-X.

Table 3-1 Present Signal Definitions

PRSNT1#	PRSNT2#	Expansion Configuration
Open	Open	No expansion board present
Ground	Open	Expansion board present, 25W maximum
Open	Ground	Expansion board present, 15W maximum
Ground	Ground	Expansion board present, 7.5W maximum.

We have never seen the present signals ever used, but we have heard of systems that will not PNP (Plug-and-Play) configure a PCI board if both the present pins are left open. We recommend installing a jumper in location 1-2 (for PRSNT2-), or 5-6 (for PRSNT1-), or both.

J31: M66EN—66MHz Enable (Pins 7–8)

The `66MHZ_ENABLE` pin (**M66EN**) indicates to the host whether the device can operate at 66 MHz or 33 MHz. Section 7.5.1 in the *PCI Specification 2.2* provides the gory details. For 33 MHz only FPGA designs, install a jumper between pins 7 and 8. For 66 MHz capable designs, leave this jumper uninstalled. Table 3-2 shows the jumper descriptions for M66EN.

Table 3-2 M66EN Jumper Descriptions

	Jumper	Description
M66EN	Installed	33 MHz
	Uninstalled	66 MHz

J31: PME–, Power Management Enable (Pin 3)

This board does not have built-in support for **PME–** (power management enable). Connecting **PME–** to an FPGA that is not powered is a bad idea—

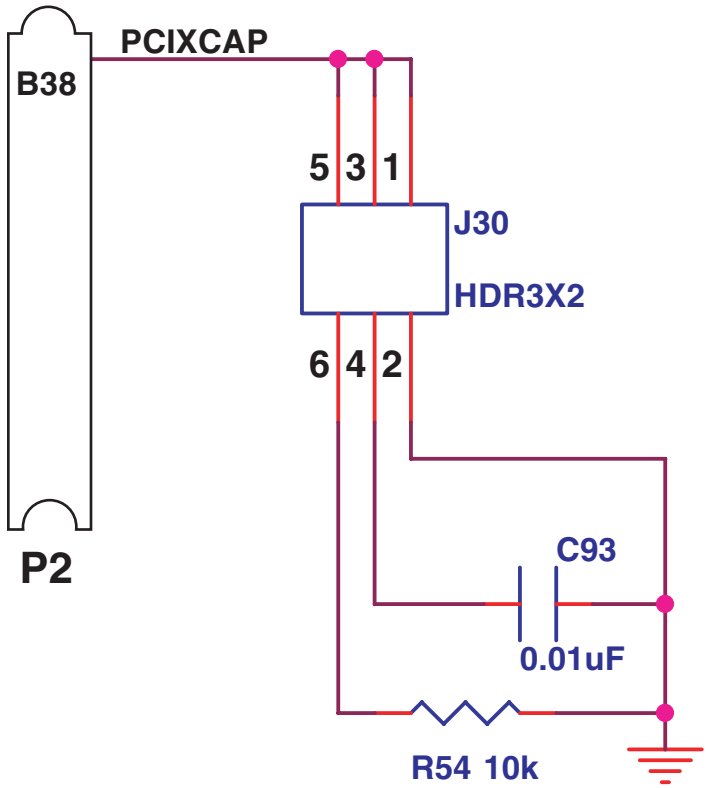


Figure 3-5 PCI-X Capability Header

the system powers up as the board is installed. **PME–** is connected to pin 3 of **J31**. This header pin allows the user to connect external circuitry to **PME–** if this functionality is desired.

J30: PCI/PCI-X Capability

Figure 3-5 shows the PCI-X Capabilities Header. Add in PCI-X boards tell the system what speed they are capable of running by the correct setting of this header.

Add-in cards indicate at which freqency they support PCI-X using a pin called **PCIXCAP**. If the card's maximum frequency is 133 MHz, this pin is left unconnected (except for a decoupling capacitor **C93**). If the card's maximum frequency is 66 MHz, it connects **PCIXCAP** to ground through a resistor **R54** (and decoupling capacitor **C93**). Conventional PCI cards connect this pin to ground.

J30—PCIXCAP

For PCI only (not PCI-X capable), jumper between pins 1 and 2.

For PCI-X 133 MHz capable, jumper between pins 3 and 4.

For PCI-X 66 MHz capable, jumper between pins 3 and 4 and pins 5 and 6.

The **PCIXCAP** jumpers are detailed in Table 3-3.

Table 3-3 PCIXCAP Jumpers

PCIXCAP	Jumper(s) Installed
PCI Only	1–2
PCI-X 133 MHz	3–4
PCI-X 66 MHz	3–4, 5–6

The M66EN and PCIXCAP Encodings are shown in Table 3-4.

Table 3-4 M66EN and PCIXCAP Encoding

M66EN	PCIXCAP	Conventional Device Frequency Capability	PCI-X Device Frequency Capability
Ground	Ground	33 MHz	Not Capable
Not Connected	Ground	66 MHz	Not Capable
Ground	Pull-down	33 MHz	PCI-X 66 MHz
Not Connected	Pull-down	66 MHz	PCI-X 66 MHz
Ground	Not Connected	33 MHz	PCI-X 133 MHz
Not Connected	Not Connected	66 MHz	PCI-X 133 MHz

Chapter 4

Clocks and Clock Distribution

Functional Overview

The DN3000k10S ASIC emulation board has a flexible and configurable clock scheme. Figure 4-1 is a block diagram showing the clocking resources and connections. .

The clocking structures for the DN3000k10S include the following features:

- 2 user-selectable socketed oscillators (**X1**, **X2**)
- 1 48 MHz oscillator (**X3**)
- 2 CY7B993 (or CY7B994) RoboclockII™ Multi-Phase PLL Clock Buffers
- 2 FCT3807 Low-Skew Clock Buffers

The Clock Grid, **J20–J22:** a 5X3 0.1 in. header distributes clock signals to two FCT3807 clock buffers and two RoboclockII™ PLL clock buffers (CY7B993 or CY7B994). The clock outputs from the buffers are dispersed throughout the board.

Two 3.3 V half-can oscillator sockets (**X1** and **X2**) and the signal CLKOUT from the CPLD provide on-board input clock solutions. The DN3000k10S is shipped with both a 14.318 MHz (**X1**) and a 100 MHz (**X2**) oscillator. Neither **X1** nor **X2** are used by the configuration circuitry, so the user is free to stuff any standard 3.3 V half-can oscillator in the **X1** and **X2** positions (more detail later in "Customizing the Oscillators" on page 4-14). The Clock Grid can also accept a 5X2 ribbon cable. This cable can provide input clocks to both of the RoboclockII's and one of the 3807 buffers.

The FCT3807 clock buffer provides a high speed 1 to 10 buffer with low skew (0.35 ns) allowing clocks A (ACLK[3:0]) and B (BCLK[3:0]) to be distributed point-to-point. The two RoboclockII's offer functional control of clock frequency and skew, among other things. The two RoboclockII™ PLL clock buffers (**U11** and **U12**) are configured via header arrays **J8**–**J10**, **J11**–**J13**, **J14**–**J16** and **J5**–**J7**. The DN3000k10S comes from the factory stuffed with CYB993V, which can operate at frequencies from 12 MHz to 100 MHz. The "CY7B" or "4V" versions of the chip are available. These chips operate from 12 MHz to 100 MHz and 24 MHz to 200 MHz respectively. Each chip has 16 output clocks along with two feedback output clocks. Two sets of eight output clocks are jumper selectable for each chip. The feedback clocks are controlled separately.

The PLL clock buffers can accept either 3.3 V LVTTL or LV Differential (LVPECL) reference inputs. The devices can operate at up to 12x the input frequency while the output clocks can be divided up to 12x the operating

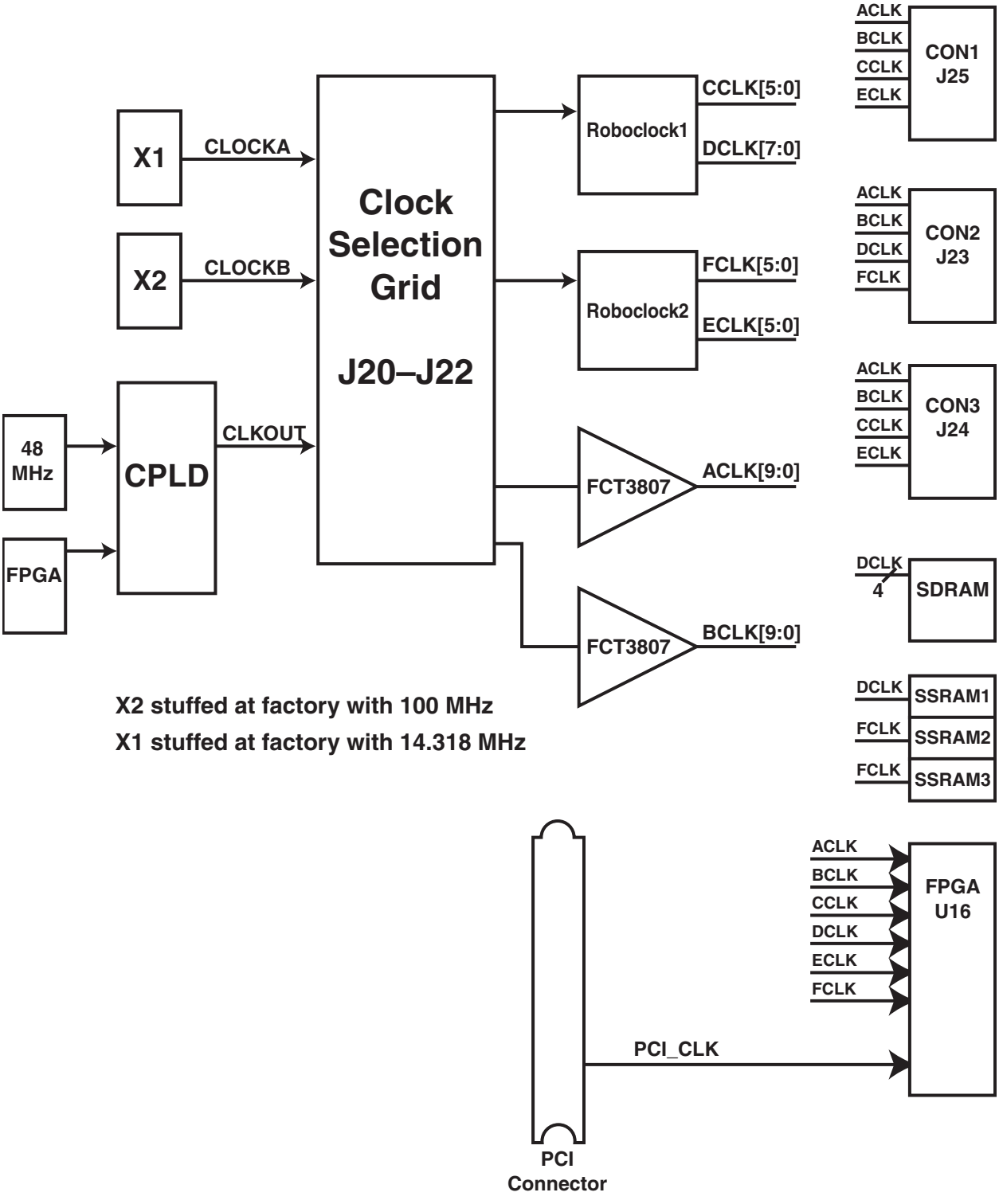


Figure 4-1 Clock Distribution Block Diagram

frequency. Phase adjustments can be made in 625 ps or 1300 ps steps up to ±10.4 ns. All adjustments are jumper selectable.

Clock Grid

Orientation and Description

The clock grid, J20–J22, gives the user the ability to customize the clock scheme on the DN3000k10S. A brief description of each pin is given in Table 4-1. The physical orientation of the pins is diagrammed in Figure 4-2. .

Table 4-1 Clock Grid Signal Descriptions

Signal	Description
CLKOUT	Clock signal from CPLD. Typically 12 MHz.
PLL1A	Input to RoboclockII #1
CLOCKA	Clock signal of oscillator #1 (X1)
BUFINB	Clock input to 3807 #1
CLOCKB	Clock signal of oscillator #2 (X2)
PLL2B_PRE	Secondary clock input to RoboclockII II #2. Differential pair with PLL2BN_PRE.
PLL2BN_PRE	Secondary clock input to RoboclockII II #2. Differential pair with PLL2B_PRE.
BUFINA	Clock input to 3807 #2
PLL1BN_PRE	Secondary clock input to RoboclockII #1. Differential pair with PLL1B_PRE.
PLL1B_PRE	Secondary clock input to RoboclockII #1. Differential pair with PLL1BN_PRE.
GND	Ground signals to provide signal integrity for ribbon cables.

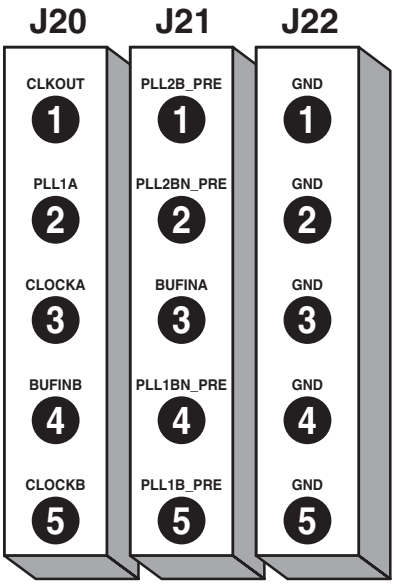


Figure 4-2 Clock Grid

Jumper Control for the Most Common Applications

Three main configurations are the most common.

First, the grid may be jumpered as follows:

Configuration #1: CLKOUT <–>PLL1A, CLOCKA <–>BUFINA and CLOCKB <–> BUFINB

Both 3807s receive their inputs from the oscillators. RoboclockII #1 receives a clock input from the CPLD. Also, RoboclockII #2 can use DCLK7 from RoboclockII #1 as an input. This is explained in "RoboclockII PLL Clock Buffers" on page 4-7. The common clock configurations are diagrammed in Figure 4-3.

Second, the input clock distribution can be configured as:

Configuration #2: CLKOUT <–>PLL2B_PRE, CLOCKA <–>PLL1A and CLOCKB <–> BUFINB

In this configuration, a 3807 #2 receives an oscillator input. RoboclockII #1 receives an oscillator input while RoboclockII #2 receives the CPLD output clock signal. 3807 #1 is unused.

Finally, the grid may be configured as:

Configuration #3: CLOCKA <–>PLL1A, and CLOCKB <–> PLL1B_PRE

Both RoboclockII's receive oscillator clock signal inputs in this case. Meanwhile, both 3807s are unused. In the last two configurations, 3807 clock driver chips are not used. The user can wire-wrap a clock to the unused driver(s) as needed. This enables full use of the timing devices on the DN3000k10S.

Also, the destination of the output clocks might dictate some other configuration. This manual and other documentation should provide

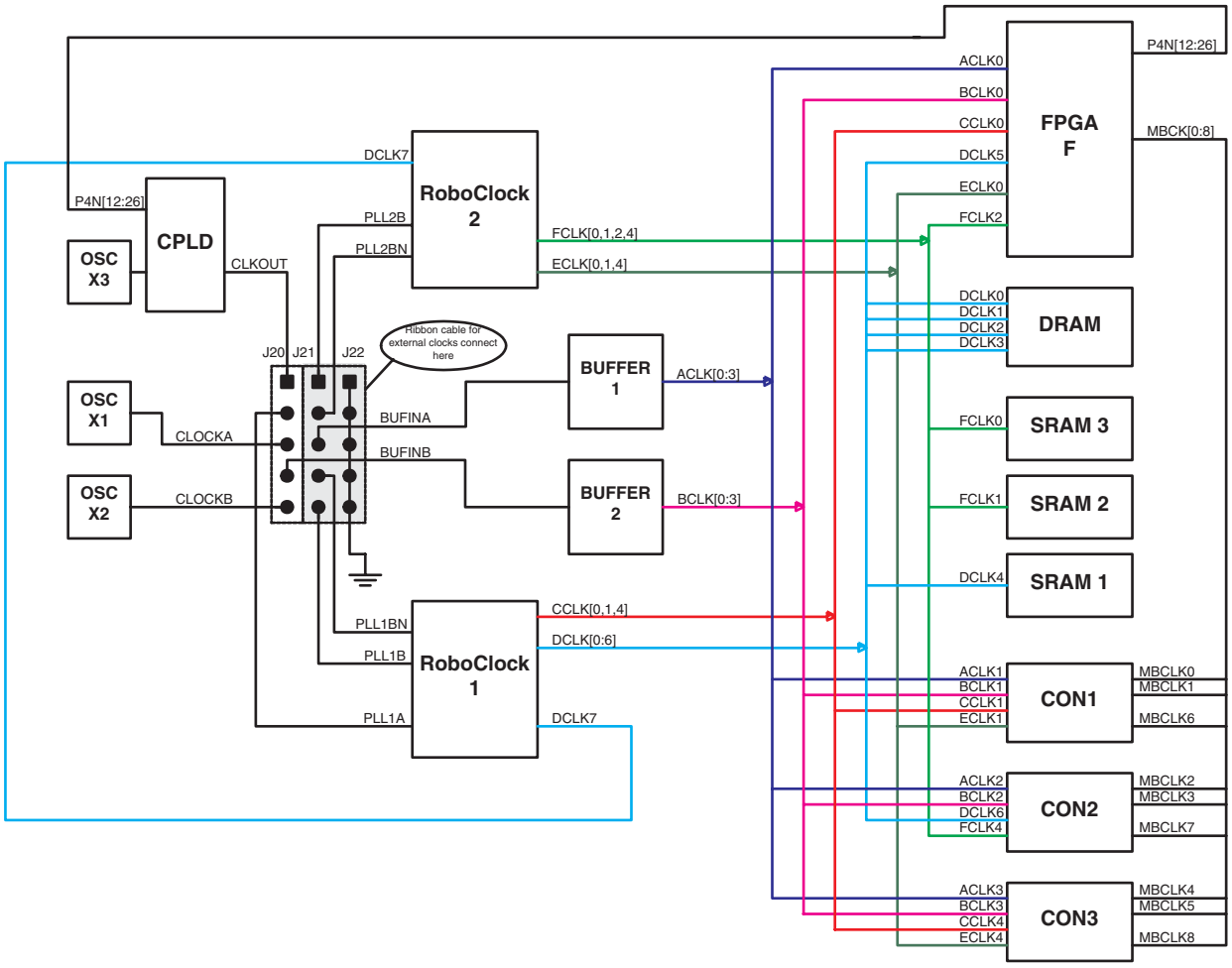


Figure 4-3 Common Clock Configurations

more than enough information to satisfy the user's needs (See Figure 4-4).

NOTE: C55, C57, C61 and C62 are stuffed with 0-ohm resistors!

Note that the schematic shows capacitors in positions **C55**, **C57**, **C61** and **C62**. The DN3000k10S has 0-ohm resistors in these capacitor positions. The termination resistors **R20**–**R23** and **R25**–**R28** are not stuffed.

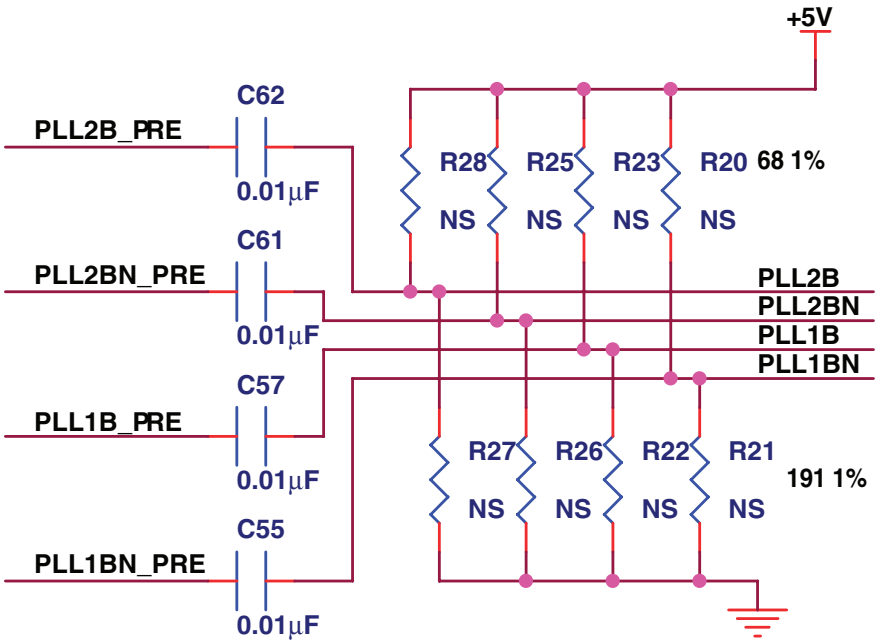


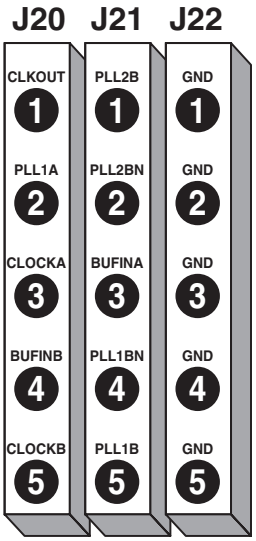
Figure 4-4 PECL Clock Input and Termination

Ribbon Cable: Providing an Off-Board Clock to the DN3000k10S

The DN3000k10S gives the user a simple means to provide off-board clocks onto the board. The user can attach 10-pin ribbon cable to J21 and J22 of the Clock Grid. J21 consists of an input to 3807 #1 and differential pair inputs to both RoboclockII's. J22 consists of ground pins for signal integrity. These signals are described in Table 4-1 on page 4-3. BUFINA is a standard 3.3 V TTL input.

Both differential pairs provide some flexibility. The user can bring a single 3.3 V TTL input. It can be attached to either input. However, the other input must be left open. The user can provide a differential clock input to the pair. The differential clock inputs must obey the electrical specifications listed in Table 4-8 on page 4-13.

While attaching a ribbon cable, the user can jumper either oscillator signal (CLOCKA, CLOCKB) to BUFINB (3807 #2) on J20. This results in full use of all of the timing devices on the DN3000k10S (See Figure 4-5).



WITHOUT 10-PIN RIBBON CABLE				
Option	**CLKOUT (J20.1) goes to:**	**CLOCKA (J20.3) goes to:**	**CLOCKB (J20.5) goes to:**	**Notes**
1	PLL2B (J21.1)	PLL1A (J20.2)	BUFINB (J20.4)	Buffer A undriven
2	PLL2B (J21.1)	BUFINB (J20.4) or BUFINA (J21.3)	PLL1B (J21.5)	Buffer A or B undriven
3	PLL1A (J20.2)	BUFINA (J21.3)	BUFINB (J20.4)	Roboclock2 is driven by Roboclock 1
4	BUFINA (J21.3)*	PLL1A (J20.2)	BUFINB (J20.4)	
5	BUFINA (J21.3)*	BUFINB (J20.4)	PLL1B (J21.5)	

*Requires wire wrap

BUFINA → Buffer A → ACLK
BUFFINB → Buffer B → BCLK
PLL1A → Roboclock1 → CCLK, DCLK (PLLSEL1 low)
PLL1B → Roboclock1 → CCLK, DCLK (PLLSEL1 high)
PLL2B → Roboclock2 → ECLK, FCLK (PLLSEL2 high)
Roboclock1 → Roboclock2 with PLLSEL2 low

WITH 10-PIN RIBBON CABLE	
Option	**(Connected to J21 and J22)**
3 external clocks	PLL1B, PLL2B and BUFINA are driven from cable. BUFINB can be jumpered to CLOCKA or CLOCKB, or left undriven.
2 external clocks	PLL2B and BUFINA are driven from cable, with PLL1A jumpered to CLKOUT or CLOCKA. Same options as above for BUFINB.
PECL clocks	The board can be set up for PECL inputs in PLL1B and PLL1BN, and in PLL2B and PLL2BN. PECL-ready boards cannot function without the cable except as in options 3 and 4 above.

Figure 4-5 External Ribbon Cable Connections

RoboclockII PLL Clock Buffers

Figure 4-6 is a functional diagram of RoboclockII 1 and RoboclockII 2.

Jumper Descriptions

Headers **J5** through **J16** are used to control the PLLs. The headers are grouped in sets of three. Of the groups of three, one header consists of GND pins. One consists of various PLL inputs. The final header consists of +3.3 V pins. The layout of the headers is shown in Figure 4-7.

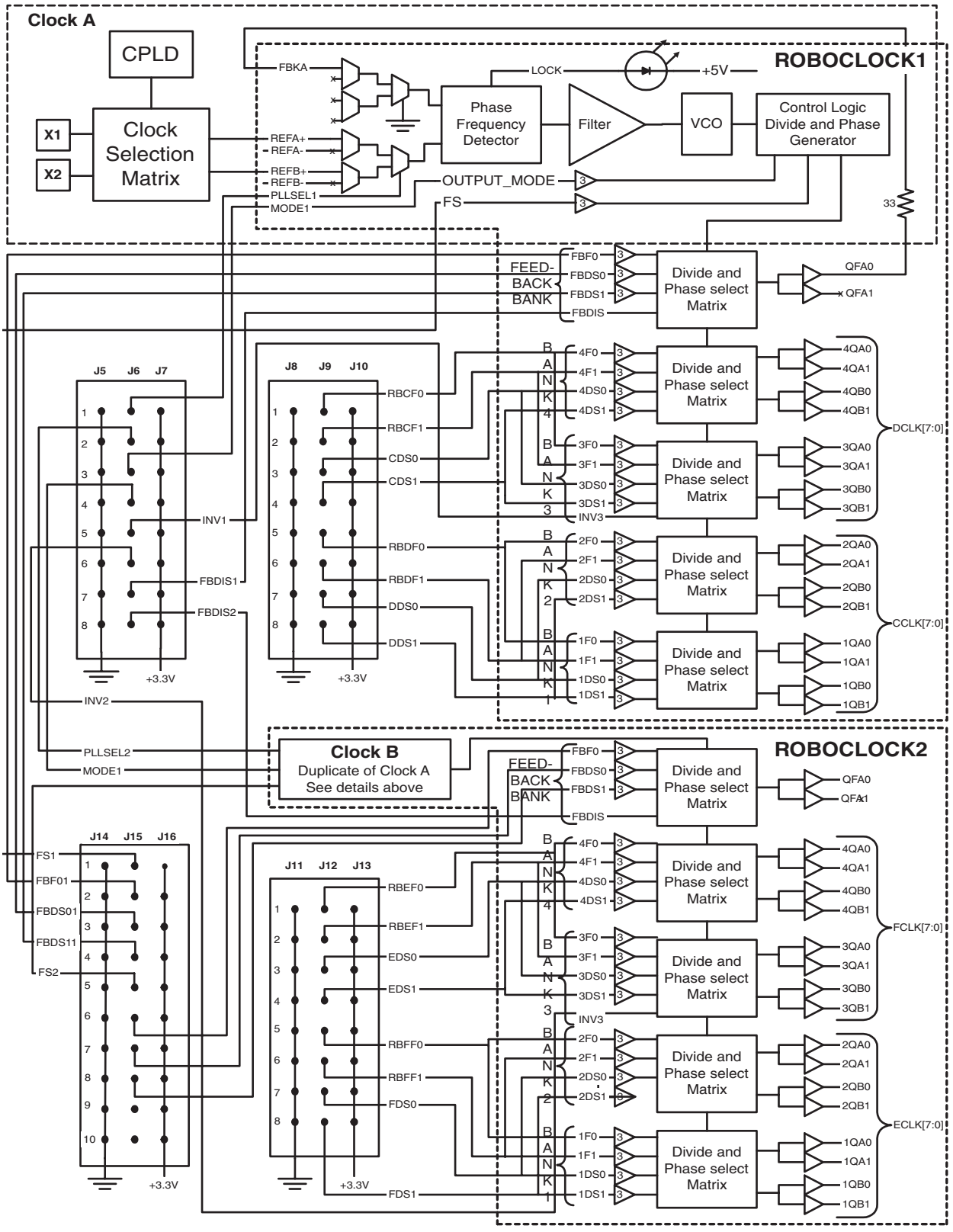


Figure 4-6 Functional Diagram of RoboclockII 1 and RoboclockII 2

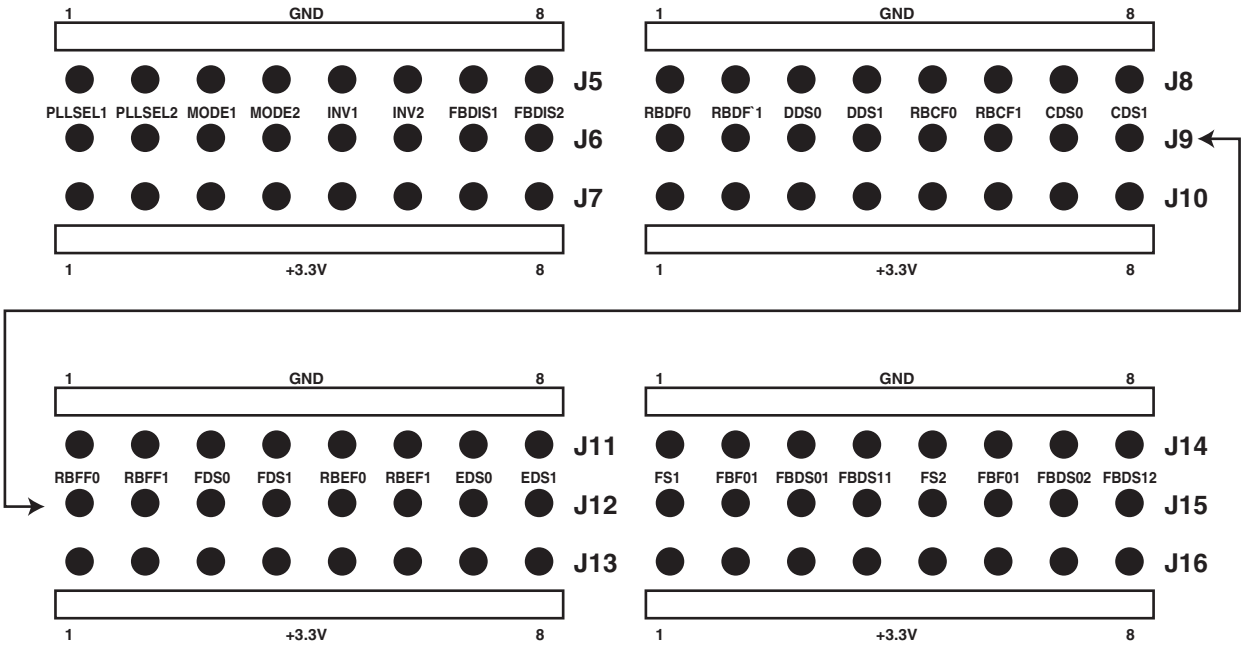


Figure 4-7 Header Layout

Note that the final two pins for header **J14**–**J16** have been omitted from the layout diagram. The Header Classifications are shown in Table 4-2.

Table 4-2 Header Classifications

Controls	GND	Input Pins	+3.3 V
Group I: General Control	J5	J6	J7
Group 2: PPL1 Divider Control	J8	J9	J10
Group 3: PLL2 Divider Control	J11	J12	J13
Group 4: Feedback and f_{NOM} Control	J14	J15	J16

The input pins are either LVTTL or 3-level input pins. The LVTTL pins need to be jumpered HIGH or LOW, which is achieved by connecting the input pin to the neighboring +3.3 V or GND pin, using a jumper. The 3-level input pins can be in a HIGH, MID, or LOW state. The HIGH and LOW states are achieved in the same way as the LVTTL pins. The MID state is reached by leaving the input pin unjumpered. The RoboclockII's have internal circuitry to bring the pin to 1.5 V when left open. The Jumper Definitions are shown in Table 4-3.

Table 4-3 Jumper Definitions

Name	Type	Default	Description
PLLSEL2[1]	LVTTL	LOW	**Input Clock Select:** If LOW, **U12**:DCLK7 (**U11**:PLL1A) is selected as the input clock. If HIGH, the **U12**:PLL2B_N (**U11**:PLL1B_N) pair is selected as the input clock.
MODE[2:1]	3-Level	HIGH	**Output Mode:** If HIGH, clock outputs disable to high-Z state. If MID, clock outputs disable to "HOLD-OFF' mode. If LOW, clock outputs disable to factory test mode.
INV2[1]	3-Level	MID	**Invert Mode:** When HIGH, clocks DCLK[3:0] (FCLK[3:0]) are inverted. When MID, these clock outputs are non-inverting. When LOW, the pairs DCLK[1:0] and DCLK[3:2] (FCLK[1:0] and FCLK[3:2]) will be complementary.
FBDIS[2:1]	LVTTL	LOW	**Feedback Disable:** When HIGH, feedback is disabled. When LOW, feedback is enabled.
RB[C-F]F[1:0]	3-Level	MID	**Output Phase Function:** Each pair controls the phase function of the respective group of outputs. See "Clock Skew" on page 4-12 for more information.
[C-F]DS[1:0]	3-Level	LOW	**Output Divider Function:** Each pair controls the divider function of the respective group of outputs. See "Clock Division" on page 4-11 for more information.
FS[2:1]	3-Level	LOW	**Frequency Select:** The input specifies the operating range of the nominal frequency (f_{NOM}). See "General Control" on page 4-11 for more information.
FBF0[2:1]	3-Level	LOW	**Feedback Output Phase Function:** The input controls the phase function of the feedback outputs. See "Feedback and Clock Multiplication" on page 4-11 for more information.
FBDS[1:0][2:1]	3-Level	MID	**Feedback Output Divider Function:** Each pair controls the divider function of the feedback outputs. See "Feedback and Clock Multiplication" on page 4-11 for more information.

General Control

FS[2:1] is a 3-Level input which determines the allowable range for the operating frequency f_{NOM} of the device. Depending on the chip grade, the PLL can operate between 12–100 MHz or 24–200 MHz. The actual f_{NOM} frequency can be determined by setting all jumpers to their defaults. Thus, f_{NOM} will be seen on all of the "divide-by-one" clock outputs. The user can set FS accordingly. The Frequency Range Settings are shown in Table 4-4.

Table 4-4 Frequency Range Settings

FS[2:1]	CY7B993V		CY7B994V	
	f_{NOM} (MHz)		f_{NOM} (MHz)	
	MIN	MAX	MIN	MAX
LOW	12	26	24	52
MID	24	52	48	100
HIGH	48	100	96	200

Feedback and Clock Multiplication

First of all, FBDIS[2:1] must be set LOW, enabling feedback. The feedback output is looped back to the feedback input. When a divided output is applied to the feedback input, the VCO (voltage controlled oscillator) of the PLL aligns the feedback input with the original input clock. Thus, with a 10 MHz input clock and the feedback outputs set to divide by 2, f_{NOM} must be 20 MHz. Consequently, 10 MHz is seen on the feedback output clocks and can be aligned with the input clocks. The feedback clock divider function actually serves as a clock multiplication mechanism for the operating frequency f_{NOM}. The divider function and the clock skew function are set in the same manner for the feedback and the normal clock outputs. See "Clock Division" on page 4-11 and "Clock Skew" on page 4-12, respectively.

Clock Division

The three pairs of DS inputs per chip are used to control the two groups of clock outputs and the feedback outputs of each PLL. The user can simply follow the Divider Function Table to acquire the desired output frequency. There are two things to remember. First, FS[2:1] must be set properly according to fNOM. Second, the FBDS feedback inputs act as operating

clock frequency multipliers. The Output Divider Settings are shown in Table 4-5.

Table 4-5 Output Divider Settings

Input Signals		Output Divider Function	
[C-fF]DS1 and FBDS1[2:1]	[C-F]DS0 and FBDS0[2:1]	Output Signals	Feedback Output Signals
LOW	LOW	/1	/1
LOW	MID	/2	/2
LOW	HIGH	/3	/3
MID	LOW	/4	/4
MID	MID	/5	/5
MID	HIGH	/6	/6
HIGH	LOW	/8	/8
HIGH	MID	/10	/10
HIGH	HIGH	/12	/12

Clock Skew

Clock skew is controlled by the "F" inputs. The clock skew may be any integer value from 0 to ±8 times the RoboclockII time unit t_U. The time unit value is derived from the operating frequency f_{NOM} and the `FS[2:1]` setting. The following equation yields the time unit t_U.

$$t_U = \frac{1}{f_{NOM} \cdot N}$$

The possible values for N are given in Table 4-6. The available skew for each RoboclockII derived clock is given in Table 4-7. Based on the following information, the user will be able to adjust the skew for any of the RoboclockII outputs.

Table 4-6 Time Unit N-factor

FS	CY7B993V		CY7B994V	
	N	f_{NOM} (MHz) at which t_U = 1 ns	N	f_{NOM} (MHz) at which t_U = 1 ns
LOW	64	15.625	32	31.25
MID	32	31.25	16	62.5
HIGH	16	62.5	8	125

Table 4-7 Clock Skew Settings

Input Signals		Output Skew Function				
RB[C-F]F1	RB[C-F]F0 and FBF0[2:1]	CCLK[3, 1:0] or ECLK[3, 1:0]	CCLK[5:4] or ECLK[5:4]	DCLK[3:0] or FCLK[3:0]	DCLK[7:4] or FCLK[5:4]	Feedback Output Signals
LOW	LOW	$-4t_U$	$-4t_U$	$-8t_U$	$-8t_U$	$-4t_U$
LOW	MID	$-3t_U$	$-3t_U$	$-7t_U$	$-7t_U$	N/A
LOW	HIGH	$-2t_U$	$-2t_U$	$-6t_U$	$-6t_U$	N/A
MID	LOW	$-1t_U$	$-1t_U$	COL1*	COL1*	N/A
MID	MID	$0t_U$	$0t_U$	$0t_U$	$0t_U$	$0t_U$
MID	HIGH	$+1t_U$	$+1t_U$	COL2**	COL2**	N/A
HIGH	LOW	$+2t_U$	$+2t_U$	$+6t_U$	$+6t_U$	N/A
HIGH	MID	$+3t_U$	$+3t_U$	$+7t_U$	$+7t_U$	N/A
HIGH	HIGH	$+4t_U$	$+4t_U$	$+8t_U$	$+8t_U$	$+4t_U$

*The clock skew is equivalent to the skew on `CCLK[3,1:0]` or `ECLK[3,1:0]`
**The clock skew is equivalent to the skew on `CCLK[5:4]` or `ECLK[5:4]`

Differential Clocks

In addition to LVTTL clock signals, the RoboclockII clock buffers can handle LV Differential (LVPECL) clocks. The user can cable in an acceptable differential signal to `PLL1B` and `PLL1BN`, or `PLL2B` and `PLL2BN` through the clock grid **J20–J22**. The signals must obey the specifications given in Table 4-8. Onboard circuitry is available to center the signals about the proper voltage, if needed.

Table 4-8 LVPECL Input Specifications

Description	Min	Max
Differential Voltage	0.4	3.3
Highest HIGH Voltage	1.0	3.3
Lowest LOW Voltage	GND	2.9
Common Mode range (crossing voltage)	0.8	3.3

The clock input of the RoboclockII can accept a superset of PECL. PECL involves a 1 V swing about $V_{CC}/2$. The RoboclockII clock input can accept a swing of up to 3.3 V about $V_{CC}/2$, which gives the user another dimension of flexibility.

The CY7B993V/4V can output LVTTL complementary (differential) signals, too. Setting INV1 (INV2) LOW will result in clocks DCLK[1:0] and DCLK[3:2] (FCLK[1:0] and FCLK[3:2]) becoming complementary pairs. A network of series and parallel resistors could be used to reduce the nominal swing of the clock signals.

Useful Notes and Hints

The CYB993V consistently outputs ~32.5 MHz signals in cases of improper settings or unacceptable clock inputs. This was observed when:

- The CY7B993V part was operating at a nominal frequency f_{NOM} of 36.4 MHz with FS set LOW.
- Identical clocks were sent to PLL2B and PLL2BN.

For the CY7B994V part, the operating frequency can reach up to 200 MHz. However, the maximum output frequency is 185 MHz. This means when 185 MHz ≤ f_{NOM} ≤ 200 MHz, the output divider must be set to at least 2. Otherwise, the RoboclockII's will output garbage.

Customizing the Oscillators

The user can customize the frequency of the clock networks by stuffing oscillators in **X1** and **X2**. The DN3000k10S is shipped with a 14.318 MHz oscillator in location **X1** and a 100 MHz oscillator in **X2**. The RoboclockII's are **not** +5 V tolerant, so +3.3 V oscillators are necessary.

NOTE: If you stuff your own oscillators, +3.3 V CMOS outputs are necessary since the RoboclockII's are not +5 V signalling tolerant!

We get our oscillators from Digi-Key (**http://www.digikey.com/**). Of note is an Epson line of oscillators called the **SG-8002 Programmable Oscillators**. Any frequency between 1.00 MHz–106.25 MHz can be procured in the normal Digi-Key shipping time of 24 hours. A half-can, +3.3 V CMOS version is needed with a tolerance of 50 ppm. The part number for an acceptable oscillator from this family would be:

SG-8002DC-PCB-ND
- package SG-531
- output enable
- 3.3 V CMOS
- 50 ppm

If the order is placed via the web page, the requested frequency to two decimal places is placed in the Web Order Notes. The datasheet is on the CD-ROM for this oscillator. The file name is SG8002DC.pdf.

Any polarity of output enabled for each oscillator (on pin 1) is acceptable. Make sure that you have the proper jumper settings at positions 9 and 10 of **J14**, **J15** and **J16**. See Figure 4-8 and Figure 4-9 for a description.

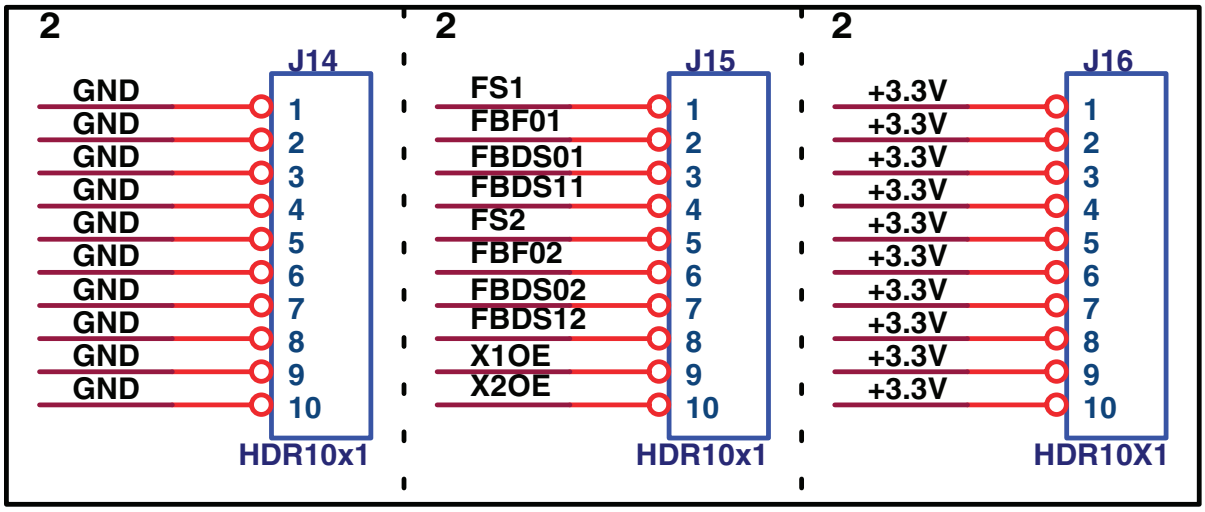


Figure 4-8 Clock OE Pin Jumper Settings

Table 4-9 Clock OE Pin Jumper Settings

Clock OE	Jumper Settings
Active High OE for X1	Jumper J15.9 to J16.9
Active Low OE for X1	Jumper J14.9 to J15.9
Active High OE for X2	Jumper J15.10 to J16.10
Active Low OE for X2	Jumper J14.10 to J15.10

Chapter 5

Memories

The DN3000k10S has four external memories: three, 36-bit SSRAMs, and one 72-bit SDRAM DIMM. The three SSRAMS are referred to as SSRAM 1 (**U18**), SSRAM 2 (**U17**), and SSRAM 3 (**U15**).

SSRAMs

The SSRAMs can be stuffed with ZBT, non-ZBT, pipeline, or flowthrough parts. We believe we have anticipated the additional address lines for the 1 M x 36 and 2 M x 36 parts when they are available. The DN3000k10S is stuffed at the factory with 512 K x 36-bit Synchronous Pipeline Burst SRAM. Samsung K7A163600M-QC1400 are probably the parts you will have stuffed into your DN3000k10S. The datasheet is on the CD-ROM in the file DS_K7A1636(18)00M.pdf. The SSRAMs are tested at 100 MHz.

SSRAM Notes

SSRAM 1 and SSRAM 3 share some I/Os with **J24**. See Figure 5-1 to figure out which ones.

SSRAM 1 is clocked by DCLK.

SSRAM 2 is clocked by FCLK.

SSRAM 3 is clocked by FCLK.

The signal connections for SSRAM 1 are shown in Figure 5-2.

The signal connections for SSRAM 2 are shown in Figure 5-3.

The signal connections for SSRAM 3 are shown in Figure 5-4.

Flowthrough SSRAMs are functionally the closest to ASIC-style memories. Pipeline SSRAMs can be clocked at faster frequencies. ZBT SSRAMs are typically one generation behind in density. The subtle differences between the styles of memories are described in the next section.

Each SSRAM position can be stuffed with a different style of SSRAM, but pin 14 is shared between the three SSRAMs. Pin 14 may be pulled high (by stuffing **R55**), pulled low (by stuffing **R56**), or left unconnected (by not stuffing **R55** or **R56**).

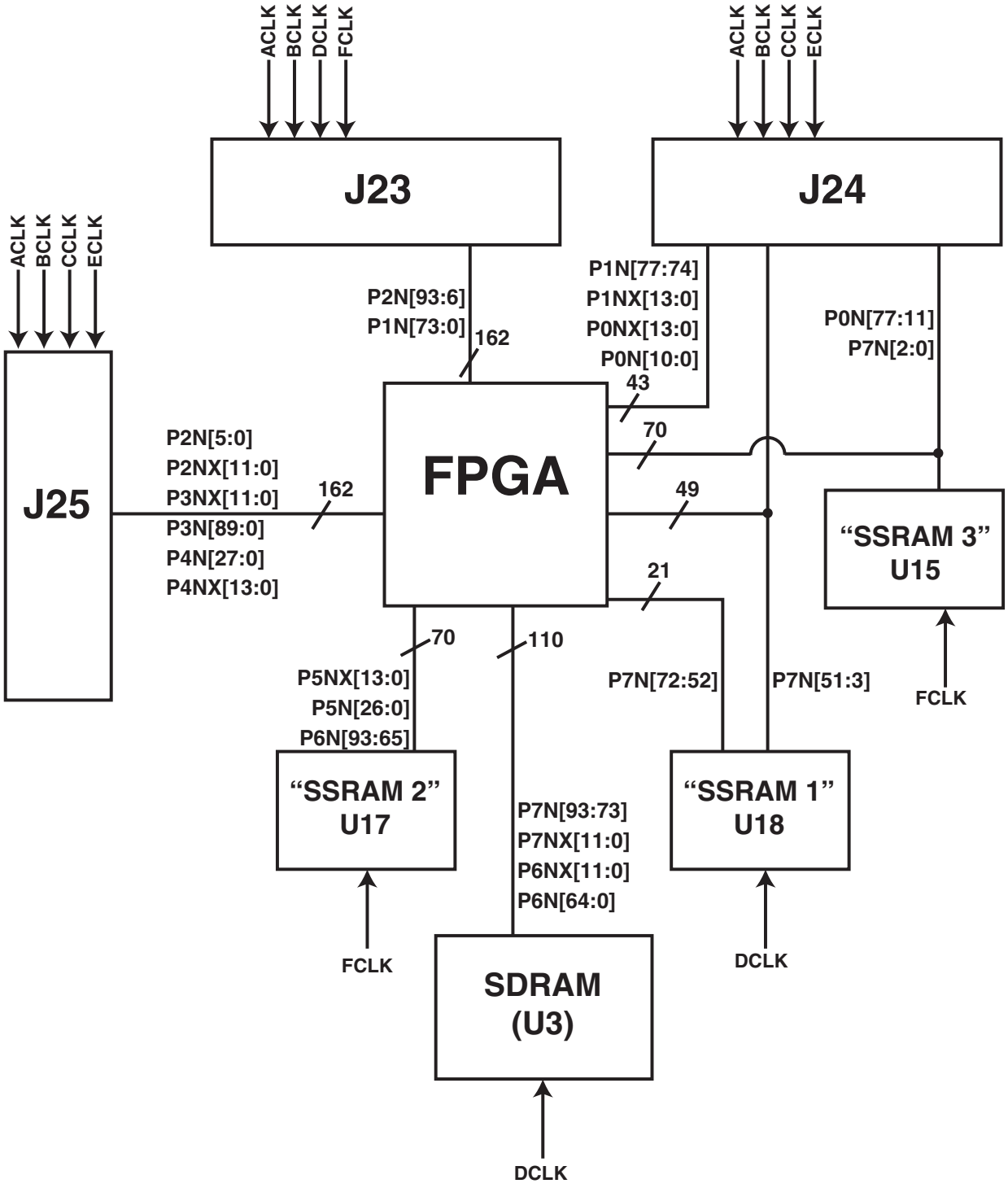


Figure 5-1 FPGA Interconnect Block Diagram

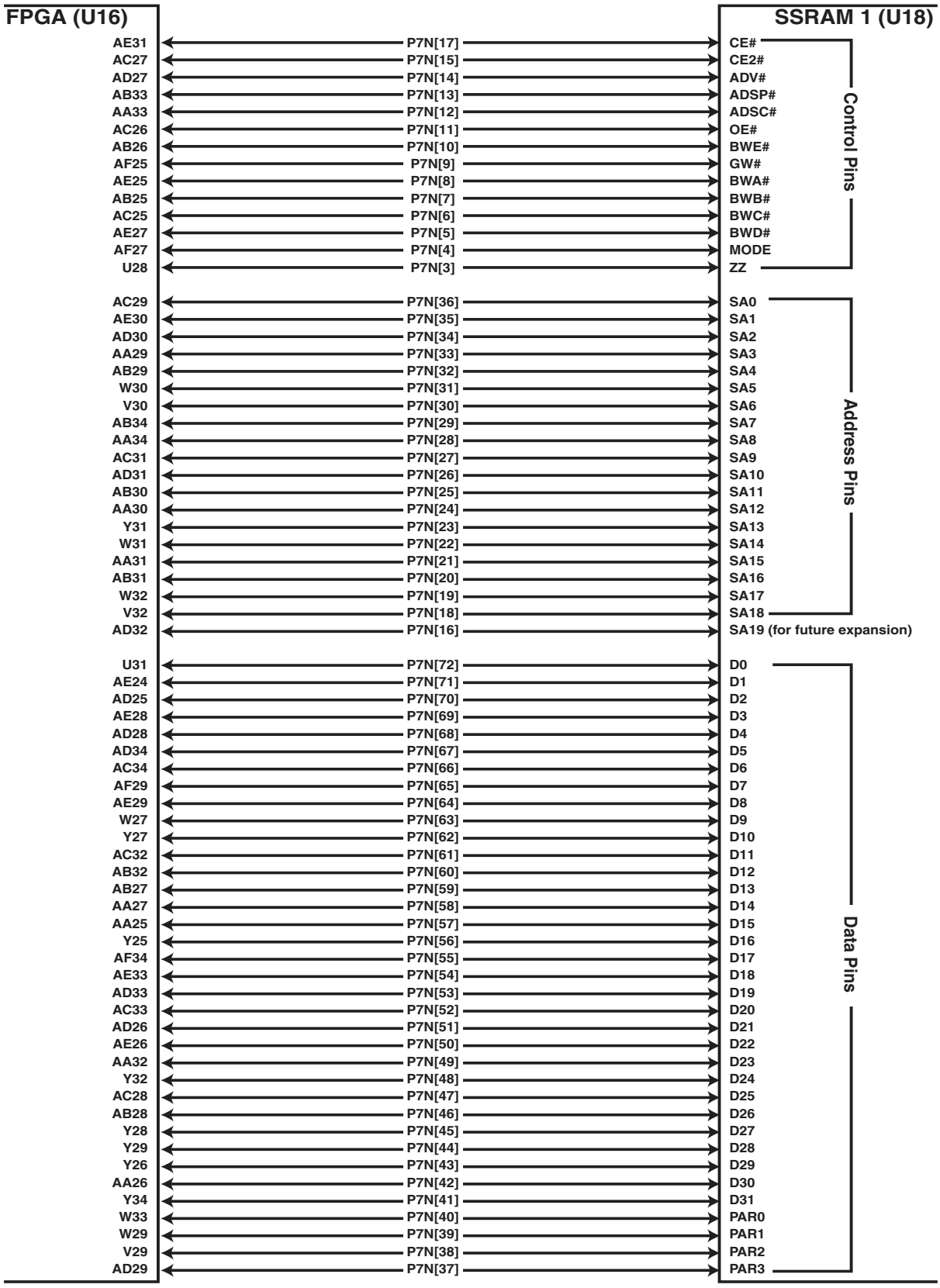


Figure 5-2 SSRAM 1 (U18) Bus Signals

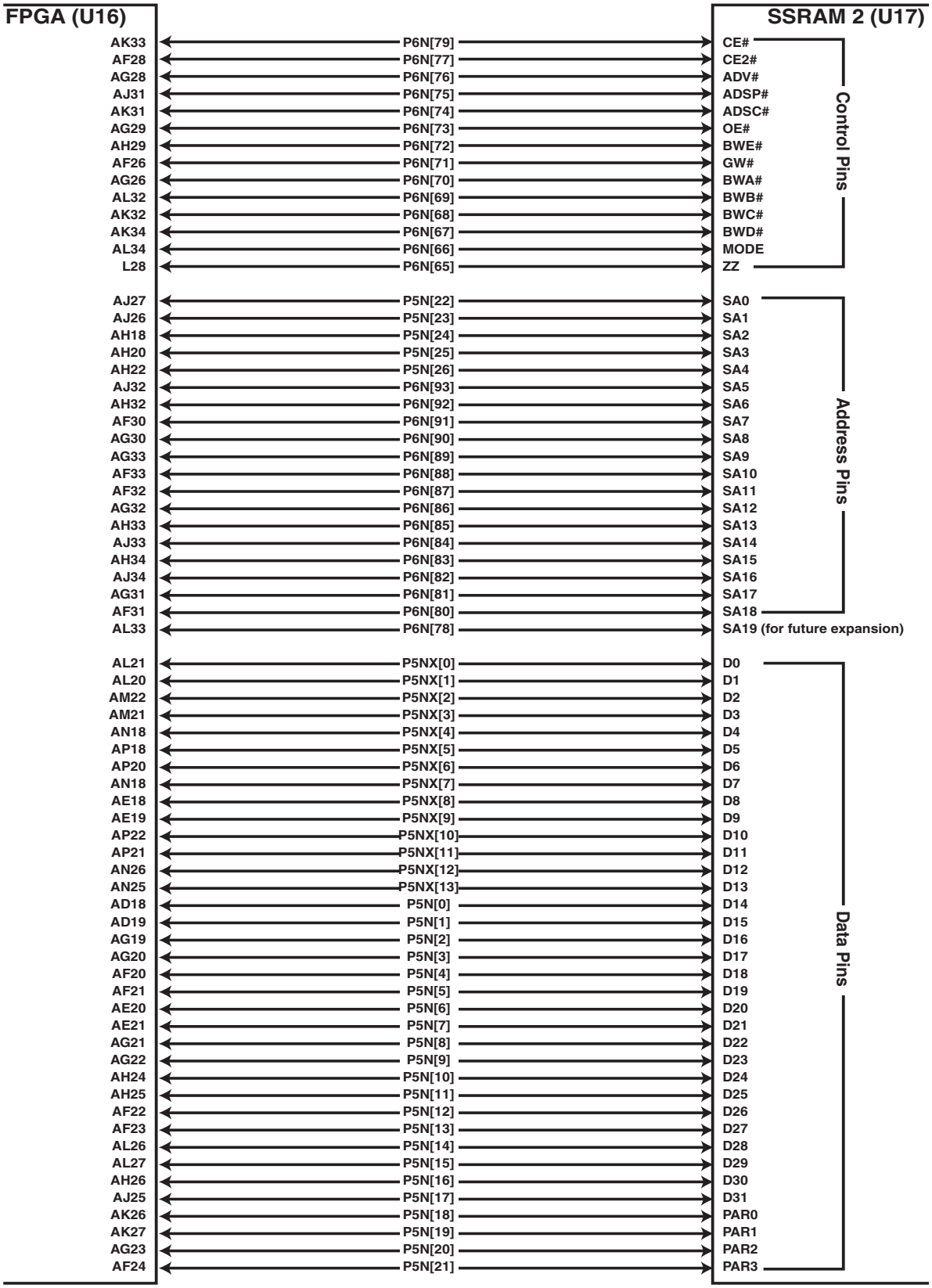


Figure 5-3 SSRAM 2 (U17) Bus Signals

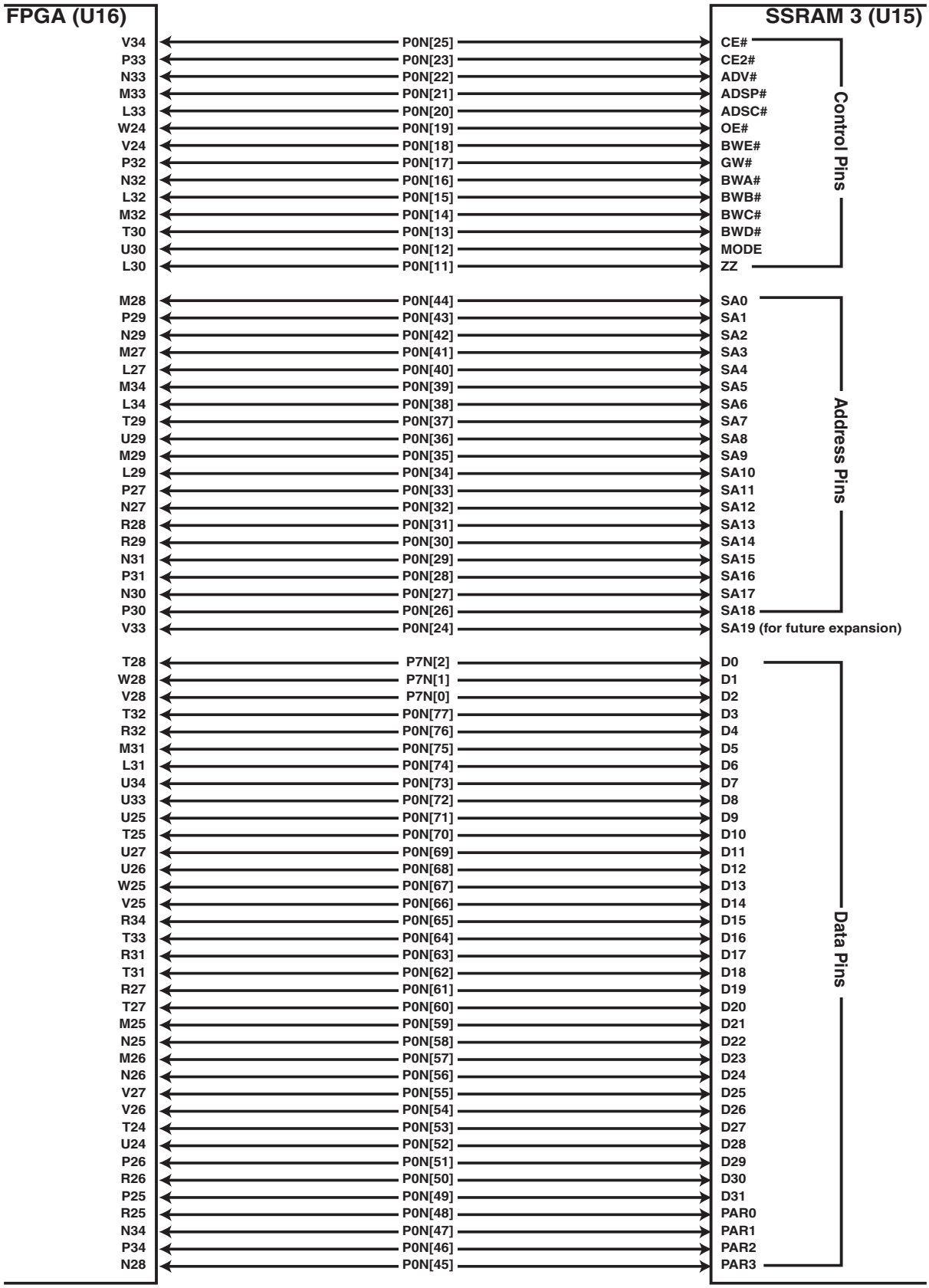


Figure 5-4 SSRAM 3 (U15) Bus Signals

Table 5-1 has the details of each SSRAM.

Table 5-1 Requirements for Non-Standard SSRAMs

	Pin 14	Pin 16	Pin 66
ZBT Pipeline AB A2D BD FED	+3.3 V Install Resistor R55 R48 R49 R90 R182	+3.3 V Short to Pin 15	+3.3 V Short to Pin 65
ZBT Flowthrough AB A2D BD FED	GND Install Resistor R56 R50 R51 R98 R181	+3.3 V Short to Pin 15	GND Short to Pin 67
Syncburst Flowthrough or Pipeline	NC No Resistors	NC No Short	NC No Short
NOTE: R55 and R56 on DN3000k10S board connect Pin 14 of all three SSRAMs together.			

Pipeline, Flowthrough, ZBT

Syncburst FT (Flowthrough) (Figure 5-5) is the most straightforward type of SSRAM available for the DN3000k10S. Write data may be accepted on the same clock cycle as the activation signal and address, and read data is returned one clock cycle after it is requested. Syncburst is designed to allow two controllers to access the same SSRAM, using two activation signals, ADSC# and ADSP#; an activation with ADSP# requires data and byte enables one clock cycle after the address and activation. Syncburst PL (Pipelined) (Figure 5-6) is identical except for registered outputs, which delay read data an additional clock cycle but may be necessary for high-speed designs.

Zero-Bus-Turnaround (ZBT) SSRAMs are designed to eliminate wait states between reads and writes by synchronizing data. Thus, ZBT FT SSRAMs

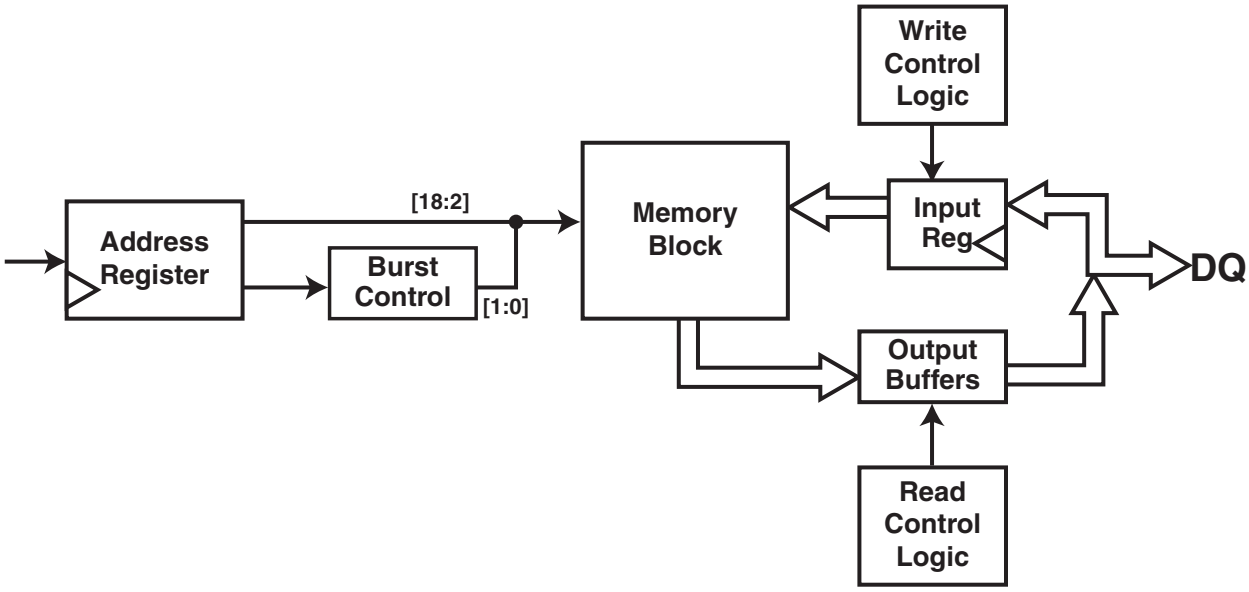


Figure 5-5 Syncburst FT

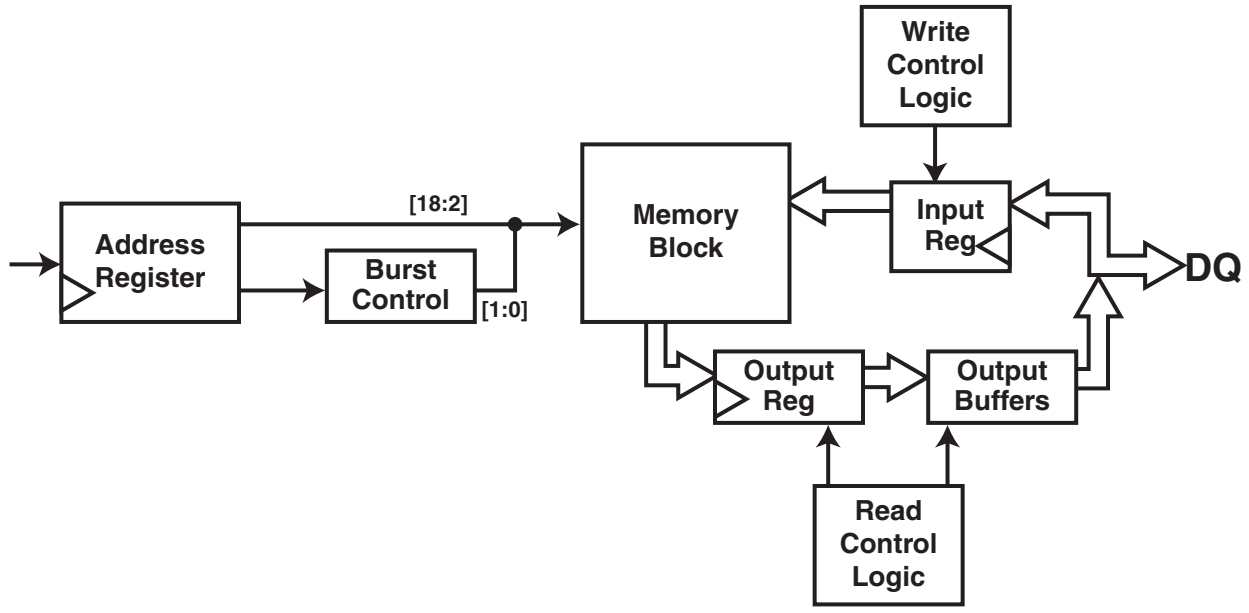


Figure 5-6 Syncburst PL

(Figure 5-7) accept and return data one clock cycle after the address phase, and ZBT PL SSRAMs (Figure 5-8) accept and return data two clock cycles after the address phase. This allows the user to begin a write burst immediately after the last word of a read burst, because read data will be returned before the first write data is required. The timing is illustrated in Figure 5-9 and Table 5-2.

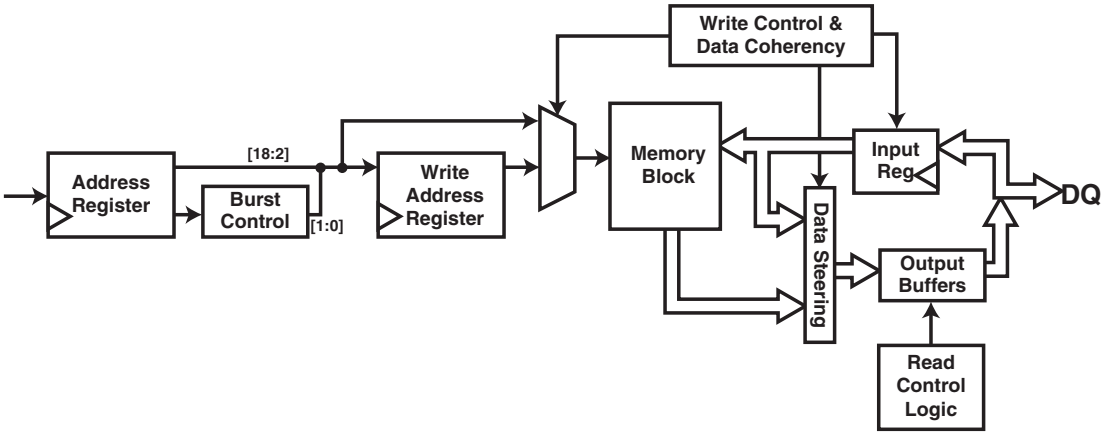


Figure 5-7 Syncburst ZBT FT

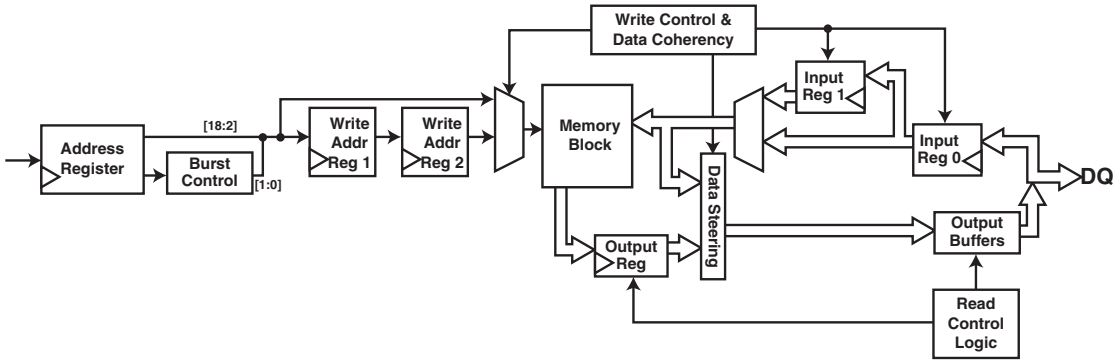


Figure 5-8 Syncburst ZBT PL

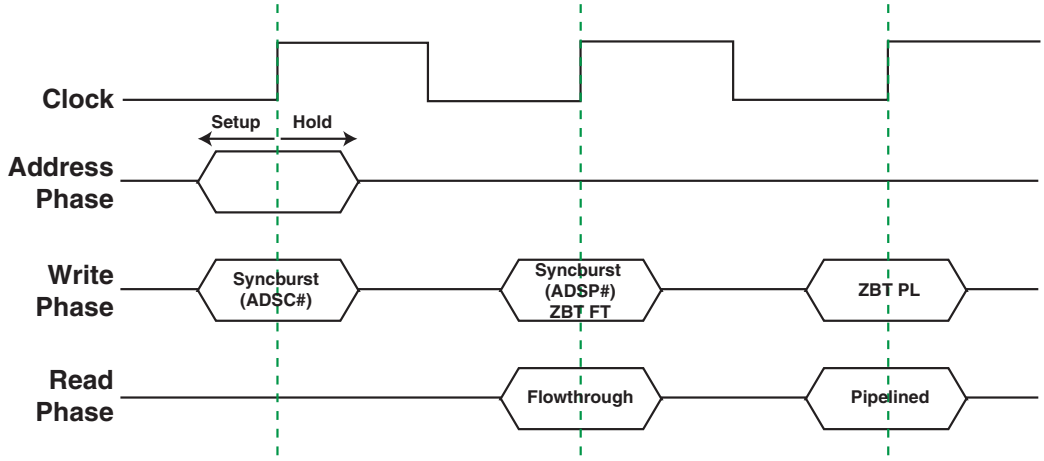


Figure 5-9 Syncburst and ZBT SSRAM Timing

Table 5-2 Syncburst and ZBT SSRAM Timing

	Syncburst	ZBT
Address Phase	CE#, CE2#, CE ADSC#, or ADSP# address or ADV#[1]	CE#, CE2#, CE R/W#, LD#[2], BWx# address or ADV[2], BWx#[3]
Write Phase	BWE#, BWx#, or GW#[4] data	data
Read Phase	Valid Data	Valid Data

[1]To continue a burst.
[2]ADV/LD# is low to load a new address, high to continue bust.
[3]For write access only.
[4]Writes to all four bytes.

SDRAM

The DN3000k10S has a socket for a +3.3 V 168-pin SDRAM DIMM. Either registered or unbuffered modules fit in the socket (**U3**). The same PC100/PC133 SDRAM modules that you put into your PC are used here. Your DN3000k10S will be stuffed and tested with a 1 Gbyte PC133 SDRAM DIMM, unless otherwise requested.

All DIMM pins are connected to the FPGA and the pins are shown in Figure 5-10 and Figure 5-11. We aren't quite sure what the largest size SDRAM DIMM is that will work in the DN3000k10S, but here is the math as best we understand it:

14 Address lines A[13:0] (multiplexed between RAS* and CAS*, address 10 not used for CAS*)	27
2 bank address BA[1:0]	2
4 chip selects (S[3:0]*) used in pairs	1

So, we think that there are 29 address bits (27 + 2) and 2 possible chips selects, which add one more address bit. This totals 30 address bits — 1 G of 72-bit long words, which is 8 Gbytes. Please tell us if this math is wrong. The +3.3 V power supply may not be able to handle an 8 Gbytes SDRAM array. A bank of this size probably consumes more than 5 A on +3.3 V.

SDRAM modules require 4 clocks — CK[3:0]. These clocks are driven by the RoboClockII 1 and the signal names are DCLK[3:0].

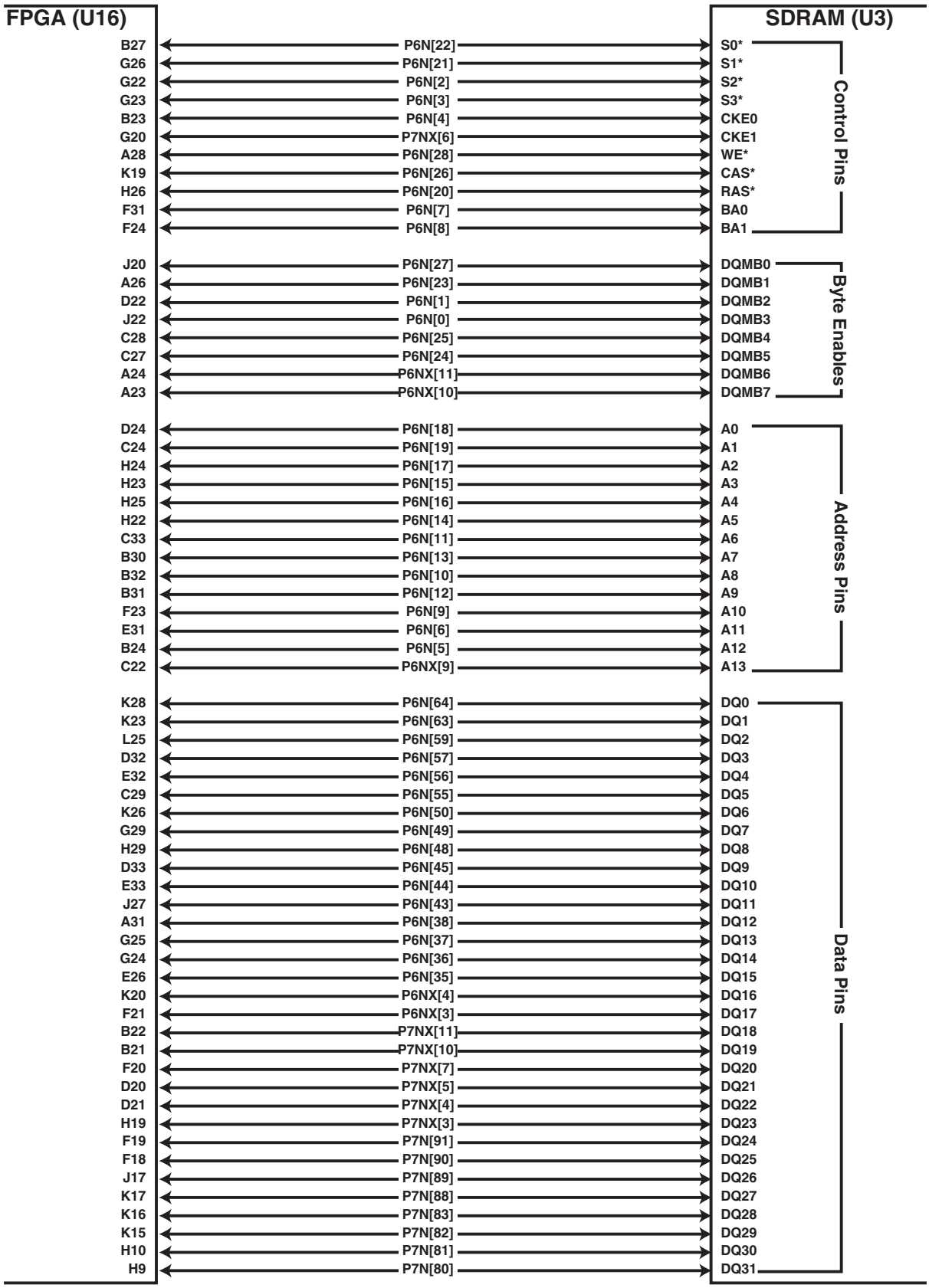


Figure 5-10 SDRAM (U3) Bus Signals (Page 1 of 2)

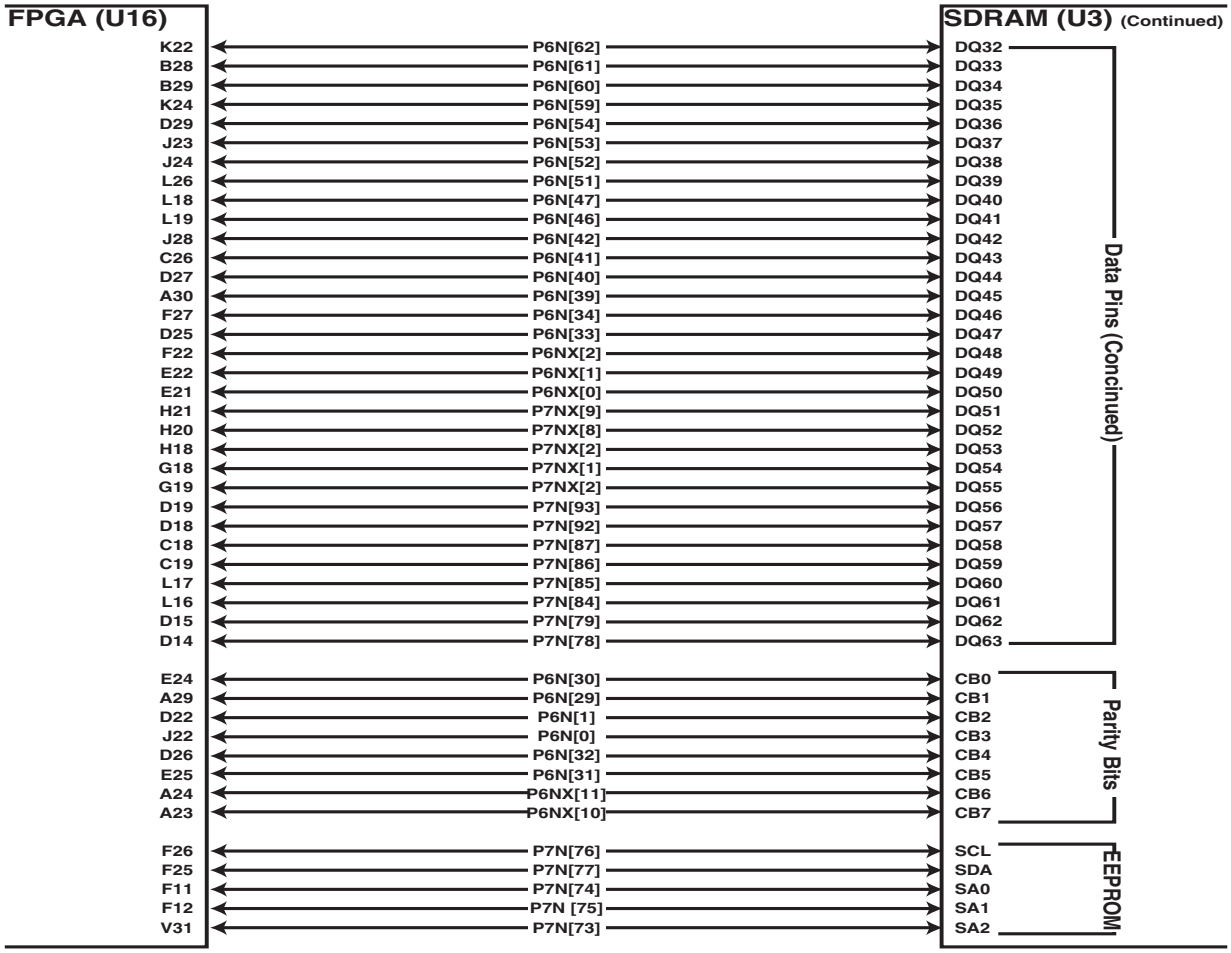


Figure 5-11 SDRAM (U3) Bus Signals (Page 2 of 2)

The CD-ROM has a datasheet of an acceptable 1 Gbyte SDRAM module from Micron. The file name is `SDF36C64_127x72G_B.pdf`.

Header Descriptions J19 and J18

Header **J18** is connected to the **WP** (Write Protect) input of the SDRAM EEPROM, and to ground. A pullup resistor keeps the **WP** signal high when the header is unconnected; adding a jumper between the two pins drives the signal low. The default configuration is no jumper.

The EEPROM holds data describing the size, configuration, and timing characteristics of the SDRAM. The data is write-protected when the **WP** signal is high. There should be little or no reason to want to overwrite the EEPROM data. Some SDRAM manufacturers simply connect the **WP** pin of the EEPROM chip to the power supply of the SDRAM, in which case **J18** has no effect whatsoever.

Header **J19** is connected to the **REGE** (Register Enable) input of the SDRAM and to ground. A pullup resistor keeps the **REGE** signal high when the header is unconnected; adding a jumper bewtween the two pins drives the signal low. The default configuration is no jumper.

On some SDRAMs, the **REGE** input may be used to select Registered or Non-Registered behavior. If **REGE** is high, the control signals will go through registers before being sent to the individual DRAMs, delaying access by one clock cycle but improving fanout; if it is low, the signals will be passed directly to the DRAMs.

The 1 GB SDRAM module shipped as part of the package is affected by both **J18** and **J19**, i.e., its EEPROM is connected to the **WP** input and it can be configured for Registered or Non-Registered behavior.

User Notes — DCMs for Clock Management on Memories

SSRAM1 and the SDRAM run on DCLK. SSRAMs 2 and 3 run on FCLK. Both clocks use an internal digital clock manager (DCM) in the reference design to reduce skew. There is also a phase shifter in the reference design, but all memories should run at the default phase shift for frequencies up to 120 MHz (note, however, that Roboclock CY7B993 is not rated for frequencies over 100 MHz).

To change the DCM phase shift by one step ($^1/_{256}$ of the clock period), PSEN must be high for one cycle of the PSCLK input. The phase will be shifted forward if PSINCDEC is high, or backward if it is low. PSDONE goes high for one cycle to indicate that the shift is finished, and PSEN must remain low until it does.

The reference design's control logic uses 8-bit registers to store phase shift information. The decimal number 128 (or hex 80) represents a phase shift of 0, i.e., the point at which the clock is de-skewed. The user may write a new number to an address in BAR0 (hexadecimal offset 000C for DCLK, 000E for FCLK), and the control logic uses the current phase shift (which can be read back at offset 000D for DCLK, 000F for FCLK) to determine whether to increment or decrement the phase shift. See Figure 5-12.

DCM Basics

The VirtexII datasheet has more information about using DCMs. There are three functions of the DCM relevant to phase issues:

1. De-Skew: The DCM may be used to de-skew a clock signal by connecting the CLK0 or CLK2X outputs to the CLKFB input. This connection is required for the other DCM functions, and it must go through a clock buffer to achieve the correct timing for the de-skew.
2. Coarse Phase Shift: Outputs CLK0, CLK90, CLK180 and CLK270 are phase shifted outputs of the same clock signal. According to Xilinx's datasheet, only four of the nine clock outputs may be used at once. The reference design uses only CLK0. (See Figure 5-13).
3. Fine Phase Shift: Inputs PSEN and PSINCDEC can be used to phase shift all DCM outputs while running. The phase shift is in units of $^1/_{256}$ of the clock period, and according to the datasheet, it is limited to no more than 2.5 ns in either direction from the de-skewed clock. (See Figure 5-14).

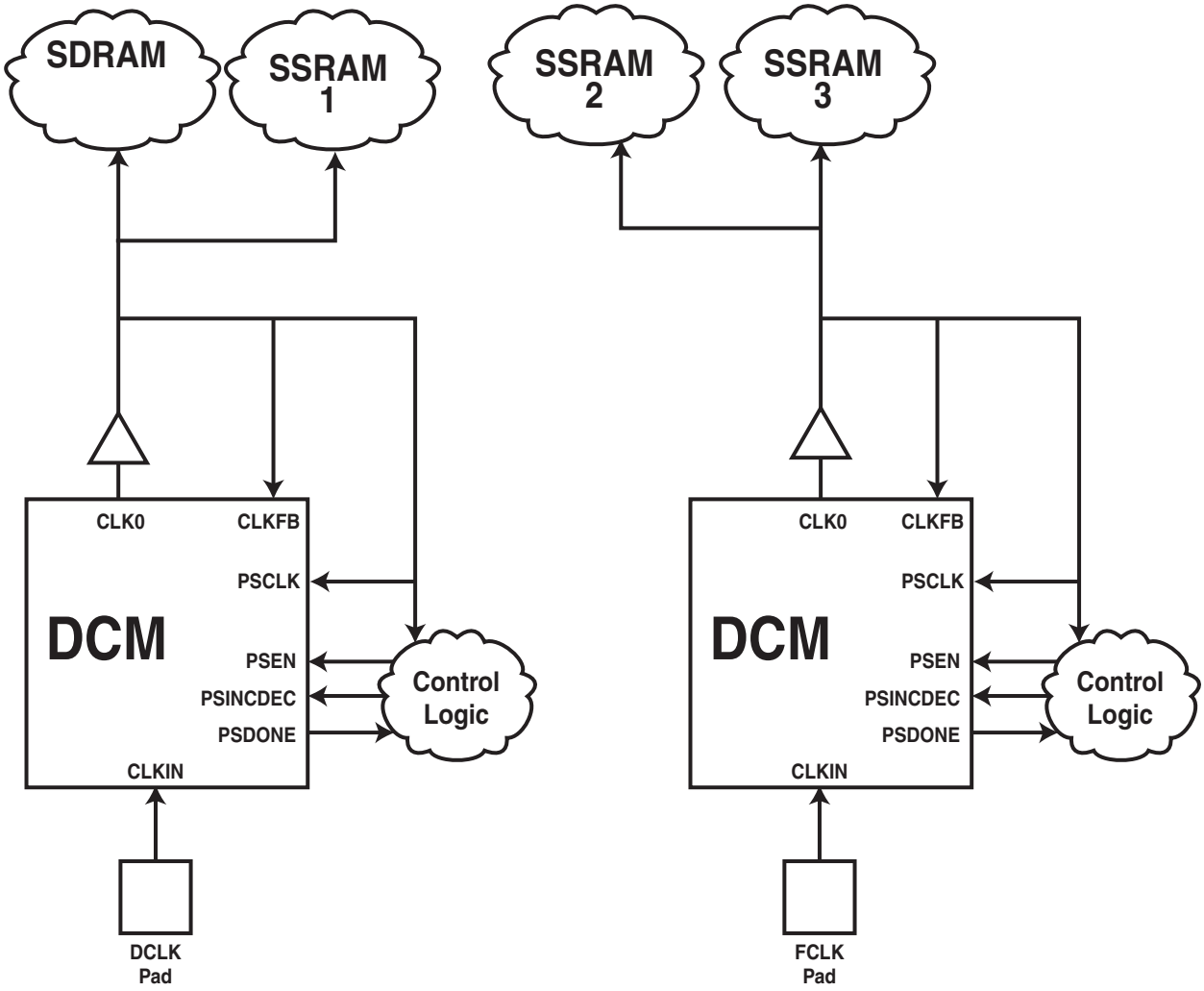


Figure 5-12 DCM Connections in Reference Design

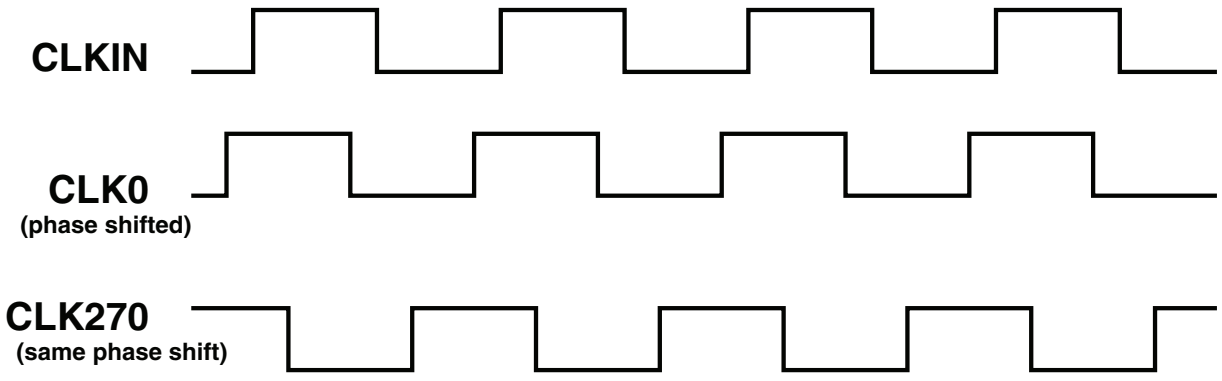


Figure 5-14 Fine Phase Shift Inputs

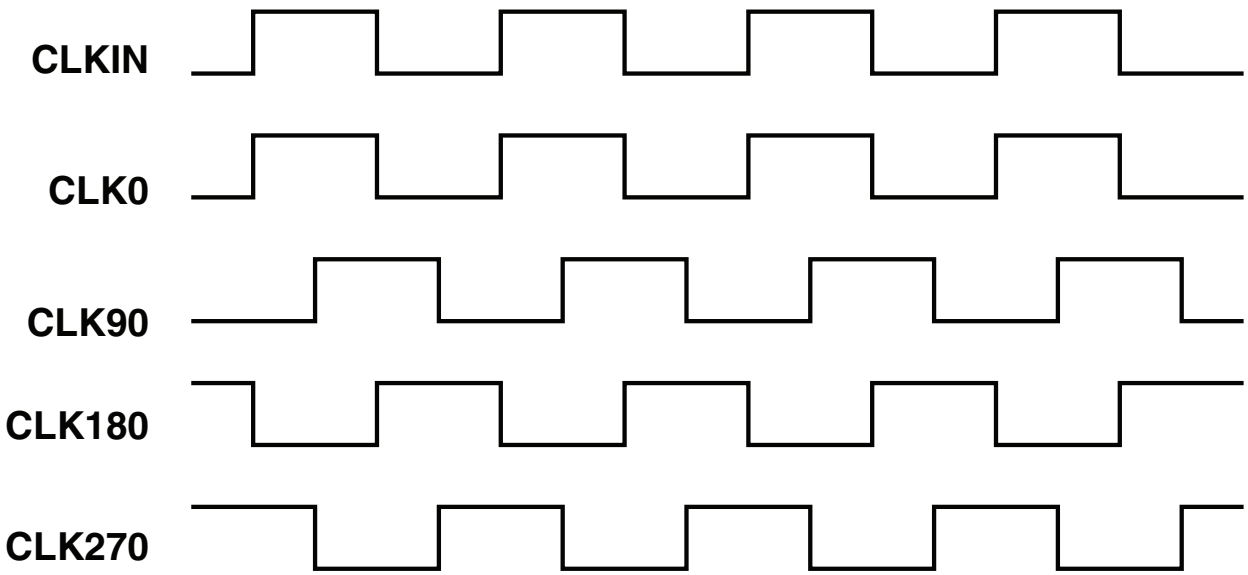


Figure 5-13 Coarse Phase Shift Outputs

Chapter 6

Power Supplies and Power Distribution

The DN3000k10S can be hosted in a +3.3 V PCI slot, or it can be used stand-alone. Figure 6-1 shows the various supplies used on the DN3000k10S and the connections of these supplies on the circuit board. The supply, +5 V, from the PCI connector (or **P1**) supplies the basic power to the DN3000k10S. The +3.3 V power from the PCI connector is ***not*** used, nor is it connected to any circuitry on the DN3000k10S.

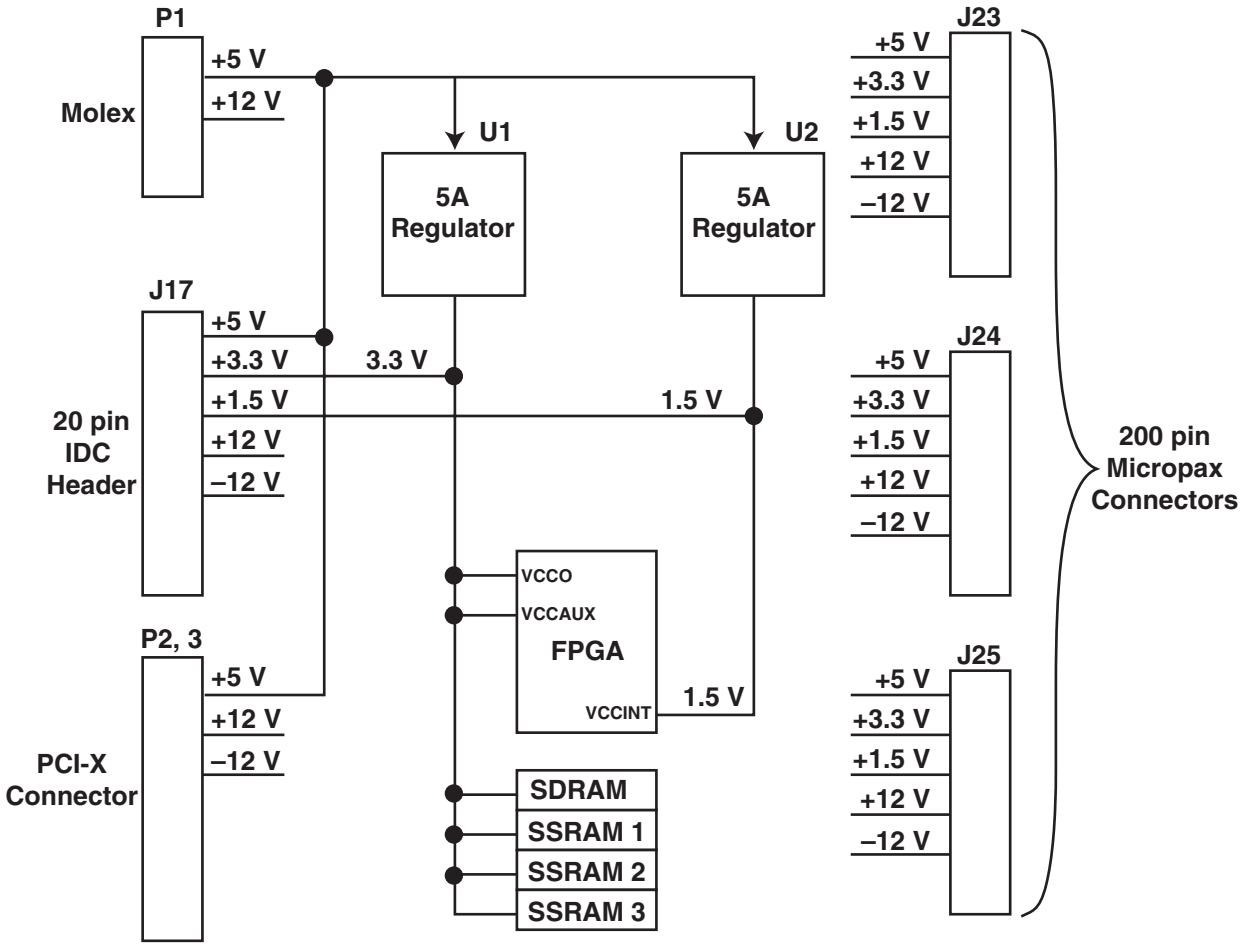


Figure 6-1 Power Distribution DN3000k10S

The DN3000k10S, when plugged into a PCI slot, has the following different power rails:

- +5 V
- +3.3 V

- +1.5 V
- –12 V
- +12 V

The power rails +3.3 V and +1.5 V are created using an LM1084 regulator with +5 V as the input. +3.3 V from the PCI fingers is ***not*** used. **U1** is for +3.3 V and **U2** is for +1.5 V. Each of these regulators has a heat sink and thermal issues were considered in the design of the PWB. Each regulator-should be able to supply the minimum 5.5 A of current without strain. The most demanding application of the DN3000k10S should fit within the 5.5 A budget on these two power rails.

The heat sinks on the regulators will get hot—possibly too hot to touch.

+3.3 V Power

The specification for the +3.3 V power is shown in Table 6-1. The +3.3 V supply is used by the following components on the DN3000k10S:

- VirtexII FPGA I/O (**U16**)
- RoboclockII's **U11**, **U12**
- 2 Clock buffers **U13**, **U14**
- CPLD (**U5**)
- Microprocessor (**U6**)
- Microprocessor SRAM (**U4**)
- 3 SSRAMs (**U15**, **U17**, **U18**)
- SDRAM DIMM (**U3**)
- 3 Oscillators **X1**, **X2**, **X3**

We do run +3.3 V a little hot. At worst case for all components, the +3.3 V power supply should never fall below +3.30 V.

Table 6-1 Specification for +3.3 V Power

	Minimum	Typical	Maximum
Voltage	+3.35 V	+3.39 V	+3.44 V
Current	N/A	N/A	12 A

+1.5 V Power

The specification for the +1.5 V power is shown in Table 6-2. The +1.5 V supply is used by the following component on the DN3000k10S:

- VirtexII FPGA V_{cc}INT (**U16**)

We also run +1.5 V a little hot. At worst case for all components, the +1.5 V power supply should never fall below +1.50 V.

Table 6-2 Specification for +1.5 V Power

	Minimum	Typical	Maximum
Voltage	+1.55 V	+1.56 V	+1.58 V
Current	N/A	N/A	12 A

If you use the DN3000k10S in a lab environment, the VirtexII FPGA will never see worst case power and temperature—so you can use typical, commercial timing.

NOTE: In a lab environment, the FPGA never sees the worst-case temperature and power. You can use typical, commercial timing!

Header J17: Off-Board Power

J17 is a power header. A standard IDC cable can be attached to this header and is useful for providing power to other circuits that may be attached to the DN3000k10S. This header is not keyed and the signals are not fused, so make sure you get pin 1 in the correct location—putting a cable on this header incorrectly will cause lots and lots of damage.

All the power rails are on this connector: +12 V, –12 V, +1.5 V, +5 V and +3.3 V. The 0.25 pins on this header are rated at more than the 3 A per pin so the cable probably limits the amount of power that you can get from this header. We did not intend for this connector to provide the power to the DN3000k10S, so don't use it for that purpose. The **J17** Power Header is shown in Figure 6-2.

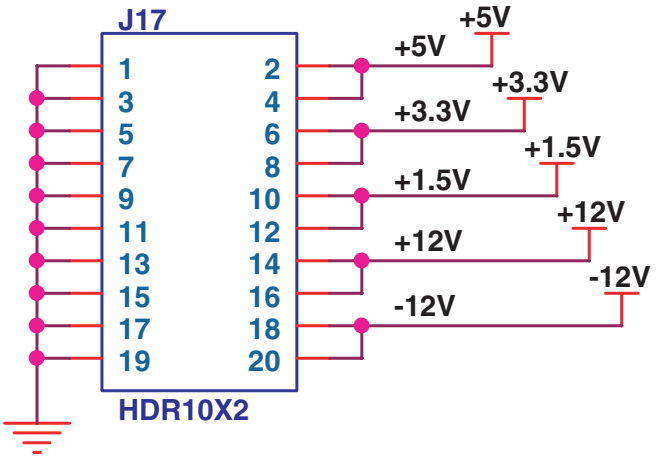


Figure 6-2 Header J17—Power

Stand-Alone Operation

The DN3000k10S can be used stand-alone, meaning it doesn't have to be plugged into a PCI slot. Connector **P1** is used to provide power to the DN3000k10S in this configuration. **P1** is a Molex drive power connector and will connect to any standard ATX power supply (see Figure 6-3). The power supply that we used is shown in Figure 6-4, but any ATX or AT style power supply will work. We use a 250-watt ATX supply. Since the DN3000K10S does not draw enough current to meet the minimums required by the supply, we plug an old disk drive into another one of the Molex connectors. The current drawn by the disk drive sinks enough current to make the switchers in the power supply happy.

The **P1** connector is rated to 13 A, far more current than the DN3000k10S can use.

The DN3000K10S, when used stand-alone, has the following different power rails:

- +5 V
- +3.3 V
- +1.5 V
- +12 V

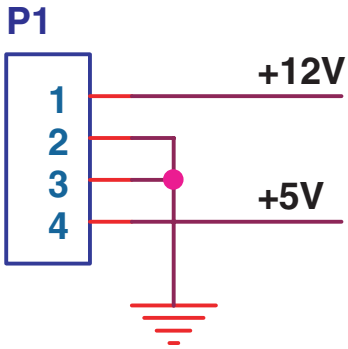


Figure 6-3 Molex Connector P1—Auxiliary Power

Figure 6-4 Example ATX Power Supply

The power rail –12 V, if needed by a daughter card, can be supplied to the DN3000k10S using the Power Header **J17**.

NOTE: If you use the DN3000k10S stand-alone with an ATX power supply, the DN3000k10S may not draw enough current to meet the minimum current required by the switchers in the supply. Connecting a disk drive to another connector will solve this problem!

By specification, a PCI board may consume a maximum of 25 watts from the fingers of the PCI connector. This power limit is below that the DN3000k10S is capable of consuming, even if daughter cards and/or large SDRAM banks are installed. The **P1** connector can be used to augment the power obtained from the PCI fingers. **P1** can be used provided that the +5 V and +12 V power rails on the connector are supplied by the same power source as the PCI fingers.

Chapter 7

Daughter Connections to DN3000k10SD—Observation Daughter Card for 200-pin Connectors

The traditional approach to experiment with new devices involving wiring together some ICs on a breadboard is fast becoming impractical and ineffective. Instead, designers using new high-density devices need custom PC boards representing a substantial investment of time and money.

Prototype boards from manufacturers can meet this demand for experimentation while eliminating the expense and time involved with custom PC boards. Additionally, such prototype boards facilitate the understanding and advantages of new device features.

Purpose

The DN3000k10SD daughter card allows external connection to the signals present on the DN3000k10S series ASIC prototyping boards. The DN3000k10S allows logic emulation with Virtex™ devices prior to committing to using them for specific applications. It allows designers to try Virtex features such as BlockRAM, DLLs, and SelectI/O™ resource, with an off-the-shelf resource.

Features

The DN3000k10S Daughter Card has the following features:

- Buffered I/O, Passive and Active Bus Drivers
- Unbuffered I/O
- Differential LVDS pairs (Note: Not available on DN3000k10S ASIC prototyping board)
- Headers for Test Points

The daughter card contains headers that may be useful with certain types of oscilloscope probes, or when wiring pins to prototype areas.

Figure 7-1 is a block diagram of the DN3000k10SD Daughter Card.

The DN3000k10SD Daughter Card is pictured in Figure 7-2.

Figure 7-3 shows the assembly drawing of the DN3000k10SD Daughter Card.

The DN3000k10SD Daughter Card provides 16 differential pairs, 48 buffered (passive/active) I/O, and 66 unbuffered I/O signals. The

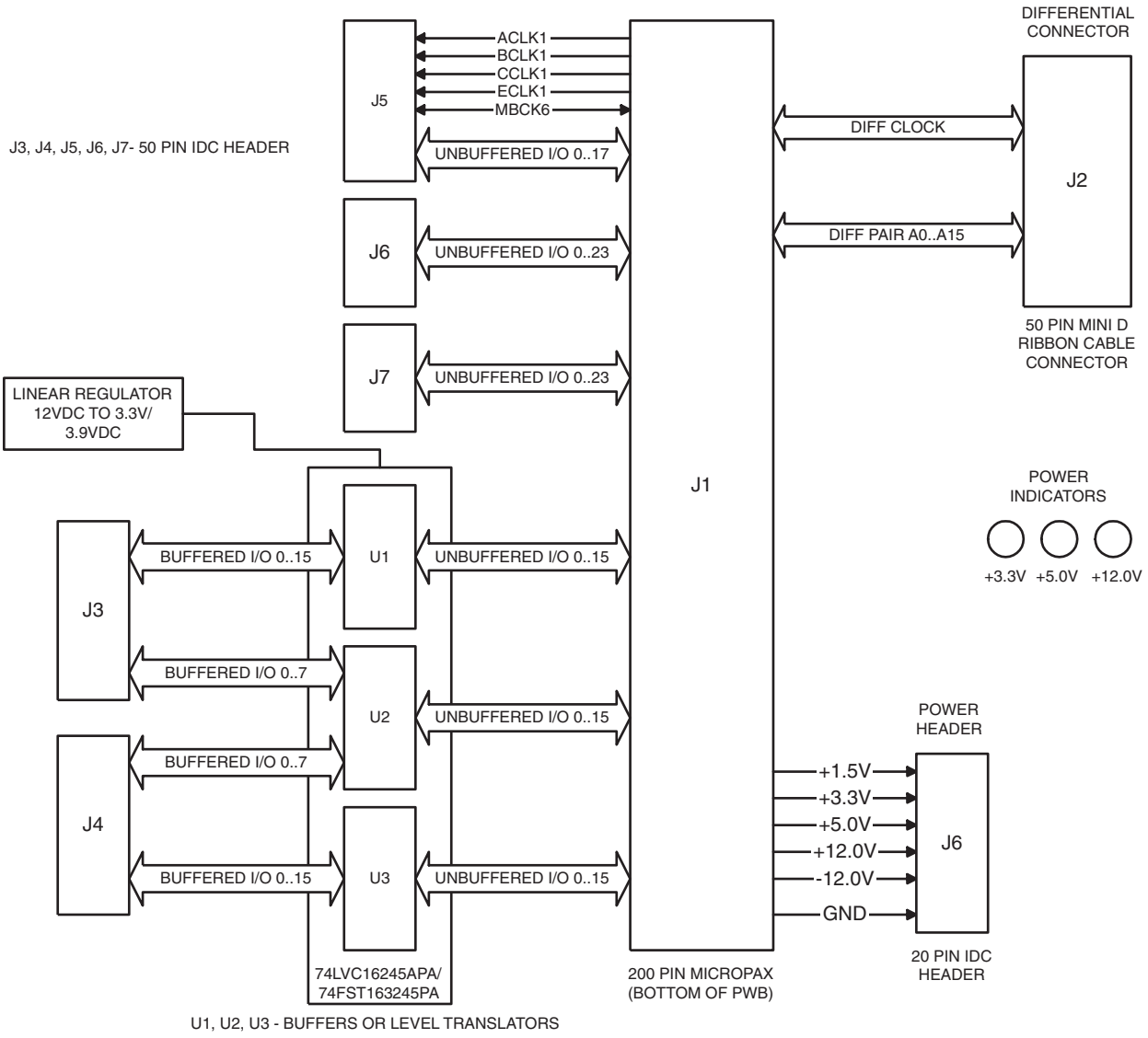
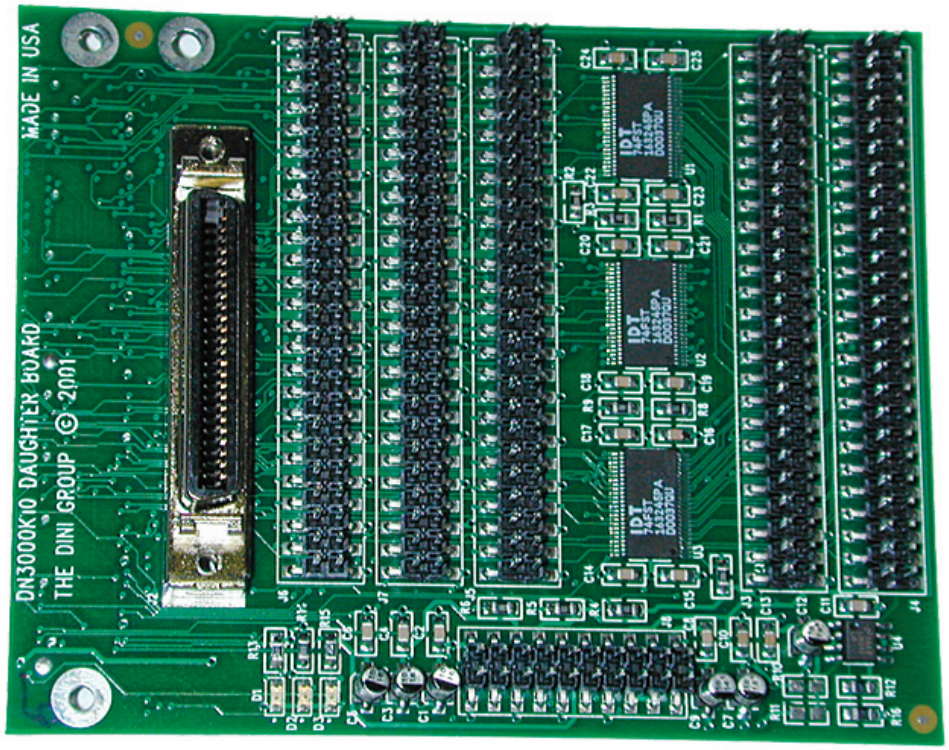
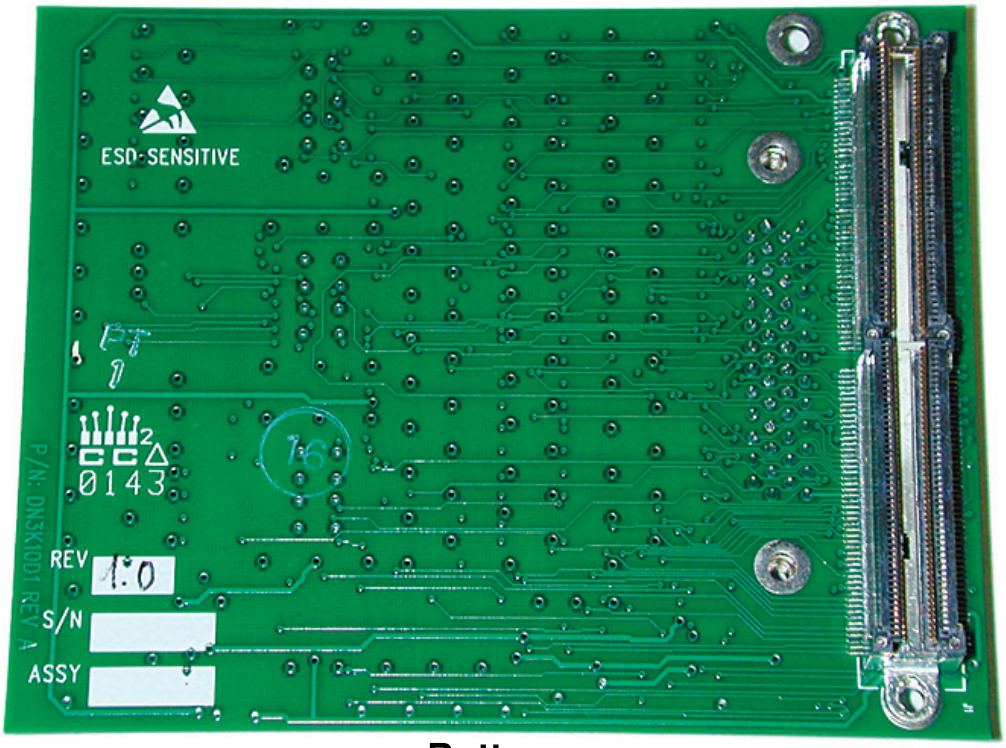


Figure 7-1 DN3000k10SD Daughter Card Block Diagram



Top



Bottom

Figure 7-2 DN3000k10S Daughter Card

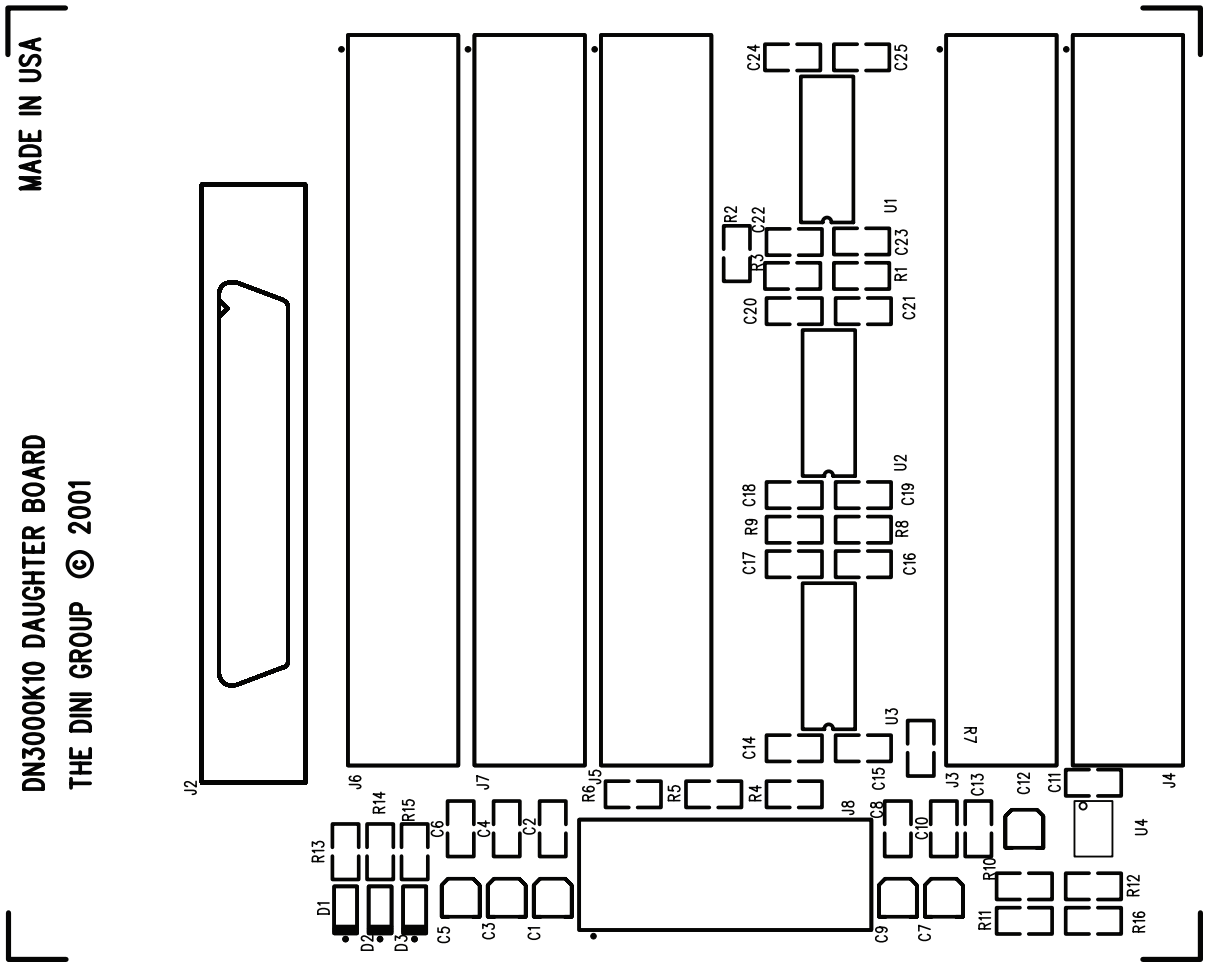


Figure 7-3 DN3000k10SD Daughter Card Assembly Drawing

IDT74FST163245 chips are used as bus switches in the passive mode, and the IDT74LVC16245A chips are used as bus transceivers in the active mode. The DN3k10D1 has separate enable/direction signals for each driver.

NOTE: Availability of these I/O signals depends on the location of the daughter card with respect to the development board.

Daughter Card LEDs

The LEDs act as visual indicators, representing the active power sources.

- **D1** — LED indicating +3.3 V present
- **D2** — LED indicating +5.0 V present
- **D3** — LED indicating +12 V present

Under normal operating conditions, all LEDs should be on.

Power Supply

A linear power supply (**U4**) is present to provide level shift/translation functions when the board is populated with bus switches.

Options

Resistors R10 and R11 can be used to select different voltage sources, +5 V or +3.3 V, respectively. When used, **U4** must be removed in order to prevent contention.

NOTE: Never populate **R10/R11** simultaneously: this will result in a shorted power supply.

Power Rating

- +5 V power supply is rated for 1 A.
- +3.3 V power supply is rated for 1 A.
- +1.5 V power supply is rated for 1 A.
- +12 V power supply is rated for 0.5 A.
- –12 V power supply is rated for 0.5 A.

Connector J8

Table 7-1 shows the connections of **J8**.

Table 7-1 Connector J8 Pins External Power

Pin	Function	Pin	Function
1	GND	11	GND
2	+5 V	12	+1.5 V
3	GND	13	GND
4	+5 V	14	+12 V
5	GND	15	GND
6	+3.3 V	16	+12 V
7	GND	17	GND
8	+3.3 V	18	–12 V
9	GND	19	GND
10	+1.5 V	20	–12 V

LVDS

Low-voltage differential signaling (LVDS) is a signaling method used for high-speed transmission of binary data over copper. It is well recognized that the benefits of balanced data transmission begin to outweigh the costs over single-ended techniques when the signal transmission times approach 10 ns. This represents signaling rates of about 30 Mbps or clock rates of 60 MHz (in single-edge clocking systems) and above. LVDS is defined in the TIA/EIA-644 standards.

NOTE: Not available on the DN3000k10 ASIC prototyping board.

Connector J2

This is a Mini D Ribbon (MDR) connector (50 pin) manufactured by 3M, used specifically for high speed LVDS signaling. The connector mates with a standard off-the-shelf 3M-cable assembly:

P/N 14150-EZBB-XXX-0LC

where XXX is:
050 = 0.5 m
150 = 1.5 m
300 = 3.0 m
500 = 5.0 m

Please contact 3M for further details: **http://www1.3m.com/**.

Unbuffered I/O

The DN3000k10SD Daughter Card provides 66 unbuffered I/O signals, including 5 single ended clock signals. The function of these signals is position dependent.

NOTE: SIgnals `P4NX7` and `P4NX6` are also used for direction select and output enable on **U2** and **U3** respectively.

Connectors J3, J4

J3, **J4**— Buffered Interface header

IDC headers (50 pin) providing 48 buffered I/O signals.

See Table 7-2 on page 7-8.

Connector J5, J6, J7

J5, J6, J7 — Unbuffered Interface Header

IDC headers (50 pin) providing 66 buffered I/O signals.

See Table 7-2 on page 7-8.

Buffered I/O

The DN3000k10SD Daughter Card provides 48 buffered I/O signals. The function of these signals is position dependent. **U1**, **U2**, and **U3** allow for different populating options, and devices can be active or passive.

Active

The LCV162245A is used for asynchronous communication between data buses. It allows data transmission from the A to the B or from the B to the A bus, depending on the logic level at the direction-control (**DIR**) input. The output-enable (**OE#**) input can be used to disable the device so that the busses are effectively isolated.

Passive

The FST163245 bus switches are used to connect or isolate two ports without providing any current sink or source capabilities. Thus, they generate little or no noise of their own while providing a low resistance path for an external driver. The output-enable (**OE#**) input can be used to disable the device so that the busses are effectively isolated.

Test Interface

The DN3000k10SD Daughter Card provides a 200-pin connector to interface to one of three test connectors on the DN3000k10 ASIC prototyping board: **J23**, **J24** and **J25**.

Connector J1

J1—Test Interface Connector

Micropax connector (200 pin) used as a standard interface to all the DINI Group development boards. This connector has a specified current rating of 0.5 amps per contact. See Table 7-2 on page 7-8.

Daughter Card I/O Connections

Table 7-2 shows the DN3000k10SD Daughter Card I/O Interconnects to connectors **J23, J24** and **J25**.

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor							
			J23, J24, or J25 Pin No.	J23		J24		J25		
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin	
1	+12 V		1	+12 V		+12 V		+12 V		
2	GND		2	GND		GND		GND		
3	ACLK[1]	J5.1	3	ACLK[2]		ACLK[3]		ACLK[1]		
4	+5 V		4	+5 V		+5 V		+5 V		
5	BCLK[1]	J5.3	5	BCLK[2]		BCLK[3]		BCLK[1]		
6	+5 V		6	+5 V		+5 V		+5 V		
7	CCLK[1]	J5.5	7	DCLK[6]		CCLK[4]		CCLK[1]		
8	GND		8	GND		GND		GND		
9	+3.3 V		9	+3.3 V		+3.3 V		+3.3 V		
10	P2N[3]	J3.1	10	P1N[71]	J10	P7N[49]	AA32	P2N[3]	G5	
11	GND		11	GND		GND		GND		
12	P2N[2]	J3.3	12	P1N[70]	H11	P7N[48]	Y32	P2N[2]	F5	
13	P2N[1]	J2.8	13	P1N[69]	C6	P7N[47]	AC28	P2N[1]	D2	
14	P2N[0]	J2.9	14	P1N[68]	D6	P7N[46]	AB28	P2N[0]	E2	
15	P2NX[7]	J3.5	15	P1N[63]	G9	P7N[41]	Y34	P2NX[7]	T5	
16	P2NX[6]	J3.7	16	P1N[62]	F10	P7N[40]	W33	P2NX[6]	U5	
17	P2NXP5[	J3.9	17	P1N[61]	A5	P7N[39]	W29	P2NX[5]	R6	
18	P2NX[4]	J3.11	18	P1N[60]	A4	P7N[38]	V29	P2NX[4]	R7	
19	P2NX[1]	J2.10	19	P1N[57]	E9	P7N[35]	AE30	PNX[1]	N1	
20	P2NX[0]	J2.11	20	P1N[56]	E8	P7N[34]	AD30	P2NX[0]	P1	
21	P3NX[9]	J2.40	21	P1N[53]	K12	P7N[31]	W30	P3NX[9]	Y1	
22	GND		22	GND		GND		GND		
23	P3NX[8]	J2.41	23	P1N[52]	J13	P7N[30]	V30	P3NX[8]	W2	
24	P3NX[5]	J3.13	24	P1N[49]	C8	P7N[27]	AC31	P3NX[5]	V11	
25	P3NX[4]	J3.15	25	P1N[48]	C7	P7N[26]	AD31	P3NX[4]	W11	
26	P3N[89]	J3.17	26	P1N[43]	A7	P7N[21]	AA31	P3N[89]	AG9	

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor							
			J23, J24, or J25 Pin No.	J23		J24		J25		
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin	
27	P3N[88]	J3.19	27	P1N[42]	A6	P7N[20]	AB31	P3N[88]	AF10	
28	P3N[87]	J3.21	28	P1N[41]	K13	P7N[19]	W32	P3N[87]	AF11	
29	P3N[86]	J3.23	29	P1N[40]	K14	P7N[18]	V32	P3N[86]	AE11	
30	P3N[83]	J3.25	30	P1N[37]	B10	P7N[15]	AC27	P3N[83]	AD10	
31	P3N[82]	J3.27	31	P1N[36]	B9	P7N[14]	AD27	P3N[82]	AE10	
32	P3N[77]	J3.29	32	P1N[31]	B12	P7N[9]	AF25	P3N[77]	AG7	
33	GND		33	GND		GND		GND		
34	P3N[76]	J3.31	34	P1N[30]	B11	P7N[8]	AE25	P3N[76]	AF7	
35	P3N[75]	J3.33	35	P1N[29]	C12	P7N[7]	AB25	P3N[75]	AJ3	
36	P3N[74]	J3.35	36	P1N[28]	C11	P7N[6]	AC25	P3N[74]	AH3	
37	P3N[69]	J2.42	37	P1N[23]	J14	P7N[1]	W28	P3N[69]	AJ1	
38	P3N[68]	J2.43	38	P1N[22]	J15	P7N[0]	V28	P3N[68]	AH1	
39	P3N[67]	J3.37	39	P1N[21]	E14	P0N[77]	T32	P3N[67]	AF9	
40	P3N[66]	J3.39	40	P1N[20]	E13	P0N[76]	R32	P3N[66]	AF9	
41	P3N[63]	J3.41	41	P1N[17]	H14	P0N[73]	U34	P3N[63]	AG5	
42	P3N[62]	J3.43	42	P1N[16]	H15	P0N[72]	U33	P3N[62]	AF5	
43	P3N[57]	J3.45	43	P1N[11]	C20	P0N[67]	W25	P3N[57]	AG2	
44	GND		44	GND		GND		GND		
45	P3N[56]	J3.47	45	P1N[10]	C21	P0N[66]	V25	P3N[56]	AF2	
46	P3N[55]		46	P1N[9]	C14	P0N[65]	R34	P3N[55]	AE9	
47	P3N[54]		47	P1N[8]	C13	P0N[64]	T33	P3N[54]	AD9	
48	P3N[49]	J4.1	48	P1N[3]	F17	P0N[59]	M25	P3N[49]	AC9	
49	P3N[48]	J4.3	49	P1N[2]	F16	P0N[58]	N25	P3N[48]	AB9	
50	P3N[47]	J2.19	50	P1N[1]	D16	P0N[57]	M26	P3N[47]	AF4	
51	P3N[46]	J2.20	51	P1N[0]	D17	P0N[56]	N26	P3N[46]	AE4	
52	P3N[43]	J4.5	52	P2N[91]	U8	P0N[53]	T24	P3N[43]	AC10	
53	P3N[42]	J4.7	53	P2N[90]	U9	P0N[52]	U24	P3N[42]	AB10	
54	P3N39	J4.9	54	P2N[87]	U6	P0N[49]	P25	P3N[39]	AD2	
55	GND		55	GND		GND		GND		
56	P3N[38]	J4.11	56	P2N[86]	T6	P0N[48]	R25	P3N[38]	AC2	

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor						
			J23, J24, or J25 Pin No.	J23		J24		J25	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
57	P3N[35]	J4.13	57	P2N[83]	R1	P0N[45]	N28	P3N[35]	AD4
58	P3N[34]	J4.15	58	P2N[82]	T2	P0N[44]	M28	P3N[34]	AC4
59	P3N[29]	J4.17	59	P2N[77]	P4	P0N[39]	M34	P3N[29]	AC6
60	P3N[28]	J4.19	60	P2N[76]	R4	P0N[38]	L34	P3N[28]	AB6
61	P3N[27]	J4.21	61	P2N[75]	N2	P0N[37]	T29	P3N[27]	AB2
62	P3N[26]	J4.23	62	P2N[74]	P2	P0N[36]	U29	P3N[26]	AA2
63	P3N[23]	J2.21	63	P2N[71]	L1	P0N[33]	P27	P3N[23]	AB5
64	P3N[22]	J2.22	64	P2N[70]	M1	P0N[32]	N27	P3N[22]	AA5
65	P3N[19]	J4.25	65	P2N[67]	R10	P0N[29]	N31	P3N[19]	AA10
66	GND		66	GND		GND		GND	
67	P3N[18]	J4.27	67	P2N[66]	P10	P0N[28]	P31	P3N[18]	Y10
68	P3N[15]	J4.29	68	P2N[63]	P6	P0N[25]	V34	P3N[15]	AB4
69	P3N[14]	J4.31	69	P2N[62]	N6	P0N[24]	V33	P3N[14]	AA4
70	P3N[9]	J2.23	70	P2N[57]	M4	P0N[19]	W24	P3N[9]	V2
71	P3N[8]	J2.24	71	P2N[56]	N4	P0N[18]	V24	P3N[8]	V1
72	P3N[7]	J4.33	72	P2N[55]	K4	P0N[17]	P32	P3N[7]	W10
73	P3N[6]	J4.35	73	P2N[54]	L4	P0N[16]	N32	P3N[6]	V10
74	P3N[3]	J4.37	74	P2N[51]	M6	P0N[13]	T30	P3N[3]	W5
75	P3N[2]	J4.39	75	P2N[50]	L6	P0N[12]	U30	P3N[2]	V5
76	P4N27	J4.41	76	P2N[47]	J2	P0N[9]	J30	P4N[27]	AD17
77	GND		77	GND		GND		GND’	
78	P4N[26]	J4.43	78	P2N[46]	K2	P0N[8]	H30	P4N[26]	AD16
79	P4N[21]	J4.45	79	P2N[41]	H3	P0N[3]	J33	P4N[21]	AE15
80	P4N[20]	J4.47	80	P2N[40]	J3	P0N[2]	K33	P4N[20]	AE14
81	P4N[19]		81	P2N[39]	H2	P0N[1]	H28	P4N[19]	AG14
82	P4N[18]		82	P2N[38]	J1	P0N[0]	J29	P4N[18]	AG13
83	P4N[13]		83	P2N[33]	H4	P0NX[9]	B18	P4N[13]	AF13
84	P4N[12]		84	P2N[32]	J4	P0NX[8]	A18	P4N[12]	AF12
85	P4N[11]		85	P2N[31]	K7	P0NX[7]	A20	P4N[11]	AL9
86	P4N[10]		86	P2N[30]	L7	P0NX[6]	B19	P4N[10]	AL10

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor							
			J23, J24, or J25 Pin No.	J23		J24		J25		
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin	
87	P4N[7]		87	P2N[27]	K6	P0NX[3]	A22	P4N[7]	AE13	
88	GND		88	GND		GND		GND		
89	P4N[6]		89	P2N[26]	J6	P0NX[2]	A21	P4N[6]	AE12	
90	P4N[3]		90	P2N[23]	F2	P1NX[13]	J34	P4N[3]	AJ9	
91	P4N[2]		91	P2N[22]	G2	P1NX[12]	H33	P4N[2]	AJ10	
92	P4NX[11]		92	P2N[17]	G6	P1NX[7]	B16	P4NX[11]	AH16	
93	+1.5 V		93	+1.5 V		+1.5 V		+1.5 V		
94	P4NX[10]		94	P2N[16]	H6	P1NX[6]	A15	P4NX[10]	AH17	
95	P4NX[7]	J7.45	95	P2N[13]	H6	P1NX[6]	A15	P4NX[10]	AJ15	
96	P4NX[6]	J7.47	96	P2N[12]	J8	P1NX[2]	D34	P4NX[6]	AH15	
97	P4NX[5]		97	P2N[11]	D1	P1NX[1]	G34	P4NX[5]	AE17	
98	P4NX[4]		98	P2N[10]	E1	P1NX[0]	F34	P4NX[4]	AE16	
99	GND		99	GND		GND		GND		
100	–12 V		100	–12 V		–12 V		–12 V		
101	GND		101	GND		GND		GND		
102	MBCK[1]	J2.27	102	MBCK[3]	E16	MBCK[5]	H16	MBCK[1]	J18	
103	+1.5 V		103’	+1.5 V		+1.5 V		+1.5 V		
104	MBCK[0]	J2.28	104	MBCK[2]	E17	MBCK[4]	H17	MBCK[0]	K18	
105	+3.3 V		105	+3.3 V		+3.3 V		+3.3 V		
106	MBCK[6]	J5.9	106	MBCK[7]	AK16	MBCK[8]	AK18	MBCK[6]	AF17	
107	GND		107	GND		GND		GND		
108	ECLK[1]	J5.7	108	FCLK[4]		ECLK[4]		ECLK[1]		
109	GND		109	GND		GND		GND		
110	GND		110	GND		GND		GND		
111	P2N[5]	J5.15	111	P1N[73]	E7	P7N[51]	AD26	P2N[5]	D3	
112	P2N[4]	J5.17	112	P1N[72]	D8	P7N[50]	AE26	P2N[4]	E3	
113	P2NX[11]	J2.2	113	P1N[67]	B5	P7N[45]	Y28	P2NX[11]	T4	
114	P2NX[10]	J2.1	114	P1N[66]	B4	P7N[44]	Y29	P2NX[10]	U4	
115	P2NX[9]	J5.19	115	P1N[65]	J11	P7N[43]	Y26	P2NX[9]	U10	
116	P2NX[8]	J5.21	116	P1N[64]	J12	P7N[42]	AA26	P2NX[8]	T10	

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor						
			J23, J24, or J25 Pin No.	J23		J24		J25	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
117	P2NX[3]	J5.23	117	P1N[59]	G11	P7N[37]	AD29	P2NX[3]	U11
118	GND		118	GND		GND		GND	
119	P2NX[2]	J5.25	119	P1N[58]	G10	P7N[36]	AC29	P2NX[2]	T11
120	P3NX[11]	J2.29	120	P1N[55]	B7	P7N[33]	AA29	P3NX[11]	AB3
121	P3NX[10]	J2.30	121	P1N[54]	B6	P7N[32]	AB29	P3NX[10]	AA3
122	P3NX[7]	J2.31	122	P1N[51]	E10	P7N[29]	AB34	P3NX[7]	Y8
123	P3NX[6]	J2.32	123	P1N[50]	E11	P7N[28]	AA34	P3NX[6]	W8
124	P3NX[3]	J5.27	124	P1N[47]	H12	P7N[25]	AB30	P3NX[3]	Y4
125	P3NX[2]	J5.29	125	P1N[46]	H13	P7N[24]	AA30	P3NX[2]	W4
126	P3NX[1]	J5.31	126	P1N[45]	C9	P7N[23]	Y31	P3NX[1]	V8
127	P3NX[0]	J5.33	127	P1N[44]	D9	P7N[22]	W31	P3NX[0]	V9
128	P3N[85]	J5.35	128	P1N[39]	A9	P7N[17]	AE31	P3N[85]	AJ5
129	GND		129	GND		GND		GND	
130	P3N[84]	J5.37	130	P1N[38]	B8	P7N[16]	AD32	P3N[84]	AH6
131	P3N[81]	J5.39	131	P1N[35]	G13	P7N[13]	AB33	P3N[81]	AK4
132	P3N[80]	J5.41	132	P1N[34]	G12	P7N[12]	AA33	P3N[80]	AJ4
133	P3N[79]	J2.3	133	P1N[33]	D11	P7N[11]	AC26	P3N[79]	AJ2
134	P3N[78]	J2.4	134	P1N[32]	D10	P7N[10]	AB26	P3N[78]	AH2
135	P3N[73]	J2.6	135	P1N[27]	F13	P7N[5]	AE27	P3N[73]	AG6
136	P3N[72]	J2.7	136	P1N[26]	F14	P7N[4]	AF27	P3N[72]	AF6
137	P3N[71]	J2.33	137	P1N[25]	D13	P7N[3]	U28	P3N[71]	AH5
138	P3N[70]	J2.34	138	P1N[24]	D12	P7N[2]	T28	P3N[70]	AG4
139	P3N[65]	J5.43	139	P1N[19]	A12	P0N[75]	M31	P3N[65]	AD8
140	GND		140	GND		GND		GND	
141	P3N[64]	J5.45	141	P1N[18]	A11	P0N[74]	L31	P3N[64]	AE8
142	P3N[61]	J5.47	142	P1N[15]	G15	P0N[71]	U25	P3N[61]	AE7
143	P3N[60]	J5.49	143	P1N[14]	F15	P0N[70]	T25	P3N[60]	AD7
144	P3N[59]	J6.1	144	P1N[13]	B14	P0N[69]	U27	P3N[59]	AG3
145	P3N[58]	J6.3	145	P1N[12]	B13	P0N[68	U26	P3N[58]	AF3
146	P3N[53]	J6.5	146	P1N[7]	C15	P0N[63]	R31	P3N[53]	AE6

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor						
			J23, J24, or J25 Pin No.	J23		J24		J25	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
147	P3N[52]	J6.7	147	P1N[6]	C16	P0N[62]	T31	P3N[52]	AD6
148	P3N[51]	J2.17	148	P1N[5]	G17	P0N[61]	R27	P3N[51]	AF1
149	P3N[50]	J2.18	149	P1N[4]	G16	P0N[60]	T27	P3N[50]	AE2
150	P3N[45]	J6.9	150	P2N[93]	V4	P0N[55]	V27	P3N[45]	AE5
151	GND		151	GND		GND		GND	
152	P3N[44]	J6.11	152	P2N[92]	U3	P0N[54]	V26	P3N[44]	AD5
153	P3N[41]	J6.13	153	P2N[89]	U2	P0N[51]	P26	P3N[41]	AB8
154	P3N[40]	J6.15	154	P2N[88]	U1	P0N[50]	R26	P3N[40]	AC8
155	P3N[37]	J6.17	155	P2N[85]	T7	P0N[47]	N34	P3N[37]	AC7
156	P3N[36]	J6.19	156	P2N[84]	U7	P0N[46]	P34	P3N[36]	AB7
157	P3N[33]	J6.21	157	P2N[81]	R3	P0N[43]	P29	P3N[33]	AD3
158	P3N[32]	J6.23	158	P2N[80]	T3	P0N[42]	N29	P3N[32]	AC3
159	P3N[31[	J2.44	159	P2N[79]	T8	P0N[41]	M27	P3N[31]	AC1
160	P3N[30]	J2.45	160	P2N[78]	R8	P0N[40]	L27	P3N[30]	AD1
161	P3N[25]	J6.25	161	P2N[73]	R9	P0N[35]	M29	P3N[25]	Y9
162	GND		162	GND		GND		GND	
163	P3N[24]	J6.27	163	P2N[72]	P9	P0N[34]	L29	P3N[24]	AA9
164	P3N[21]	J2.29	164	P2N[69]	N3	P0N[31]	R28	P3N[21]	AB1
165	P3N[20]	J6.31	165	P2N[68]	P3	P0N[30]	R29	P3N[20]	AA1
166	P3N[17]	J6.33	166	P2N[65]	N5	P0N[27]	N30	P3N[17]	AA8
167	P3N[16]	J6.35	167	P2N[64]	P5	P0N[26]	P30	P3N[16]	Y7
168	P3N[13]	J6.37	168	P2N[61]	P8	P0N[23]	P33	P3N[13]	AA6
169	P3N[12]	J639	169	P2N[60]	N8	P0N[22]	N33	P3N[12]	Y6
170	P3N[11]	J2.47	170	P2N[59]	L2	P0N[21]	M33	P3N[11]	Y3
171	P3N[10]	J2.48	171	P2N[58]	M2	P0N[20]	L33	P3N[10]	W3
172	P3N[5]	J6.41	172	P2N[53]	L3	P0N[15]	L32	P3N[5]	W7
173	GND		173	GND		GND		GND	
174	P3N[4]	J6.43	174	P2N[52]	M3	P0N[14]	M32	P3N[4]	V7
175	P3N[1]	J6.45	175	P2N[49]	M7	P0N[11]	L30	P3N[1]	W6
176	P3N[0]	J6.47	176	P2N[48]	N7	P0N[10]	K29	P3N[0]	V6

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN3000k10 I/O Connec tor						
			J23, J24, or J25 Pin No.	J23		J24		J25	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
177	P4N[25]	J7.1	177	P2N[45]	K5	P0N[7]	G30	P4N[25]	AG16
178	P4N[24]	J7.3	178	P2N[44]	L5	P0N[6]	F30	P4N[24]	AG15
179	P4N23	J7.5	179	P2N[43]	N9	P0N[5]	F33	P4N[23]	AF15
180	P4N[22]	J7.7	180	P2N[42]	M9	P0N[4]	G33	P4N[22]	AF14
181	P4N[17]	J7.9	181	P2N[37]	M8	P0NX[13]	J31	P4N[17]	AH12
182	P4N[16]	J7.11	182	P2N[36]	L8	P0NX[12]	H31	P4N[16]	AH13
183	P4N[15]	J7.13	183	P2N[35]	F1	P0NX[11]	G32	P4N[15]	AK11
184	GND		184	GND		GND		GND	
185	P4N[14]	J7.15	185	P2N[34]	G1	P0NX[10]	F32	P4N[14]	AK10
186	P4N[9]	J7.17	186	P2N[29]	H5	P0NX[5]	J32	P4N[9]	AG12
187	P4N[8]	J7.19	187	P2N[28]	J5	P0NX[4]	H32	P4N[8]	AG11
188	P4N[5]	J7.21	188	P2N[25]	N10	P0NX[1]	B26	P4N[5]	AH10
189	P4N[4]	J7.23	189	P2N[24]	M10	P0NX[0]	B25	P4N[4]	AH9
190	P4N[1]	J7.25	190	P2N[21]	F3	P1NX[11]	A14	P4N[1]	AM6
191	P4N[0]	J7.27	191	P2N[20]	G3	P1NX[10]	A13	P4N[0]	AL6
192	P4NX[13]	J7.29	192	P2N[19]	L9	P1NX[9]	K31	P4NX[13]	AL14
193	P4NX[12]	J7.31	193	P2N[18]	L10	P1NX[8]	K30	P4NX[12]	AL15
194	P4NX[9]	J7.33	194	P2N[15]	J7	P1NX[5]	A17	P4NX[9]	AN17
195	GND		195	GND		GND		GND	
196	P4NX[8]	J7.35	196	P2N[14]	H7	P1NX[4]	B17	P4NX[8]	AP17
197	P4NX[3]	J7.37	197	P2N[9]	E4	P1N[77]	B3	P4NX[3]	AP15
198	P4NX[2]	J7.39	198	P2N[8]	F4	P1N[76]	C2	P4NX[2]	AN16
199	P4NX[1]	J7.41	199	P2N[7]	K9	P1N[75]	K27	P4NX[1]	AN9
200	P4NX[0]	J7.43	200	P2N[6]	J9	P1N[74]	J26	P4NX[0]	AN10

Chapter 8

Reset Schemes, LEDs, Bus Bars and 200 Pin Connectors

Reset Schemes

A LTC1326 chip from Linear Technology controls reset functionality for the DN3000k10S. Figure 8-1 shows the distribution of the reset signal PWRRST–. In addition to controlling the reset, the power supplies rails +5 V, +3.3 V, and +1.5 V are threshold detected by the LTC1326. Under-voltage conditions will case the assertion of the reset signal.

The LTC1326 has a push-button. Momentarily depressing this button causes a 200 ms reset pulse on the signal PWRRST–. If the push-button is depressed for 2 seconds and held, PWRRST– is asserted continuously. LED5, when lit, means that reset is asserted, so if you press and hold **S1**, you should see LED5 illuminate after a few seconds. If LED5 illuminates for any reason during normal operation, this indicates that PWRRST– is active and that something is wrong. Note that if you press **S1** and release it quickly, you probably won't see LED5 since the 200 ms reset pulse is not strong long enough for the eye to observe.

Depressing the push-button S1 causes the following sequence of events:

1. Reset of the CPLD and µP
2. FPGA configuration is cleared
3. If the jumpers on **J2** are set for SelectMAP and there is a valid SmartMedia card inserted into the socket, then the FPGA will be configured. A SmartMedia card is valid if it complies with the SSFDC specification and contains a file named `main.txt` in the root directory. If the card is invalid or there is no card present, then the FPGA will not be configured.
4. The Main Menu will appear.

The identical sequence of events occurs at power-up.

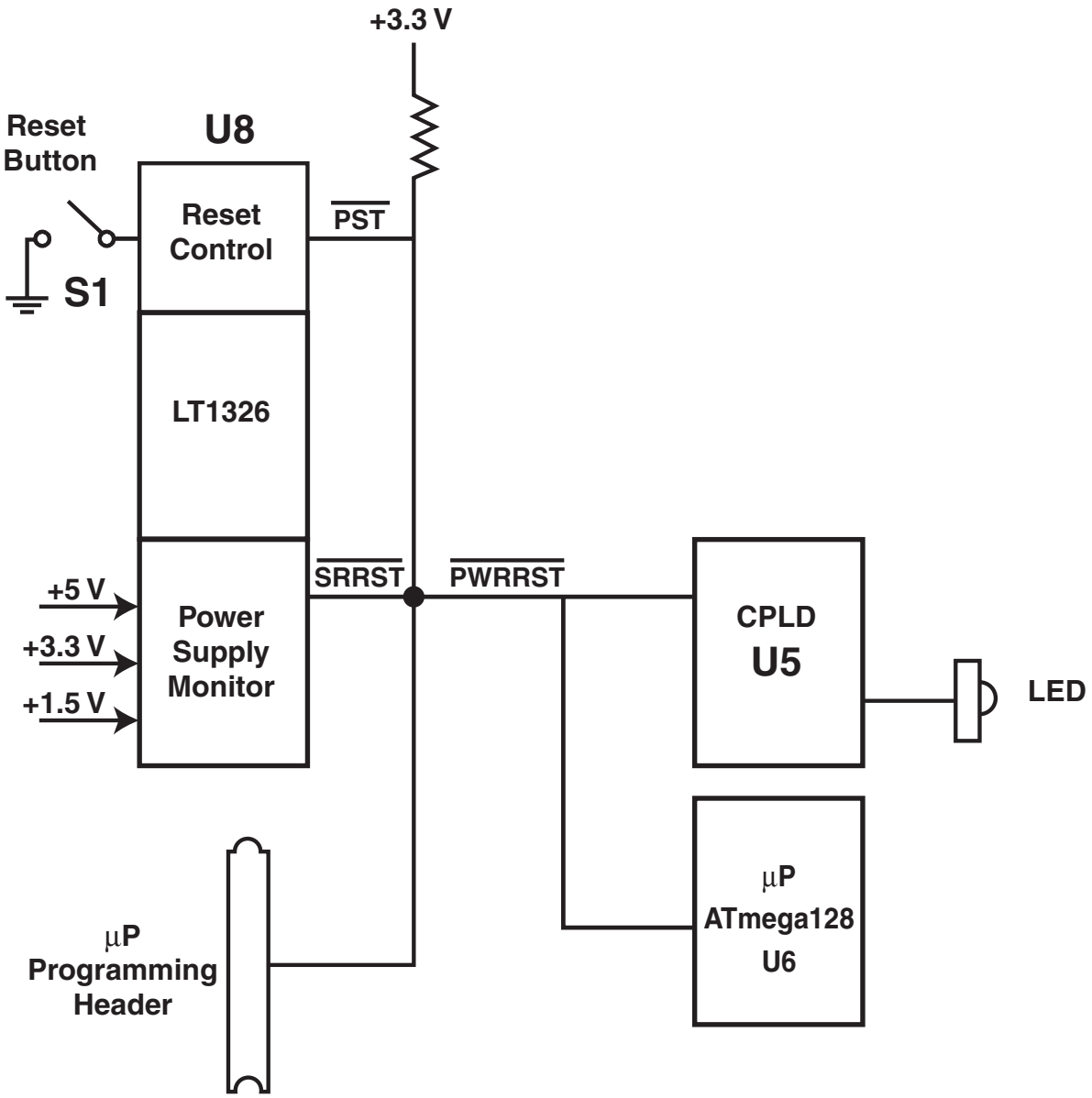


Figure 8-1 Reset Functionality

LEDs

The DN3000k10S has eight LEDs that are used to visually communicate the status of circuitry (Figure 8-2).

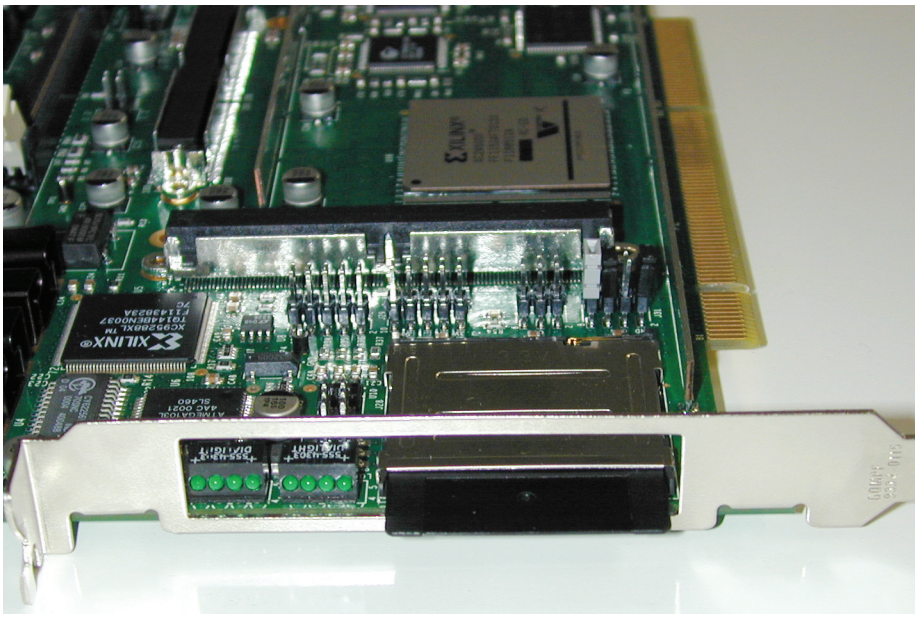


Figure 8-2 DN3000k10S LEDs

From left to right, the LEDs are labeled: LED0, LED1, LED2, LED3, LED4, LED5, LOCK1, LOCK2 (see Figure 8-3).

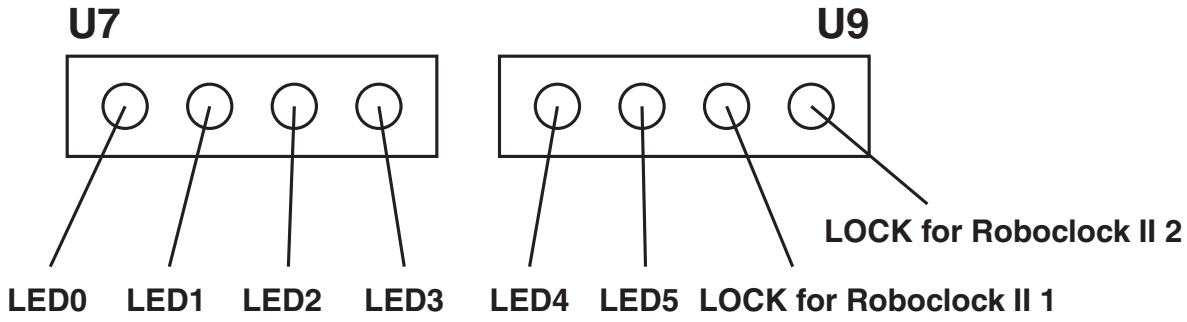


Figure 8-3 DN3000k10S LED Diagram

The LEDs have the following functions:

LED0	Lights when the configuration from the SmartMedia was successful.
LED1	Lights when there is not a valid SmartMedia card detected.
LED2	Lights when there is a configuration error.
LED3	Lights on data transfer from SmartMedia to FPGA.
LED4	Lights when FPGA is not configured (when DONE is not high).

LED5 Lights on reset (**S1** must be down for at least 3 seconds for LED1 to light).

LOCK1 Lights when the PLL in RoboclockII 1 is LOCKED

LOCK2 Lights when the PLL in RoboclockII 2 is LOCKED

You are free to reprogram the CPLD to use any or all of the LEDs for your own purposes.

J26 — LED Signals Header

The header J26 (Figure 8-4) contains the LED signals for oscilloscope observations and debugging. You can also route FPGA signals through the CPLD to this header for debug purposes.

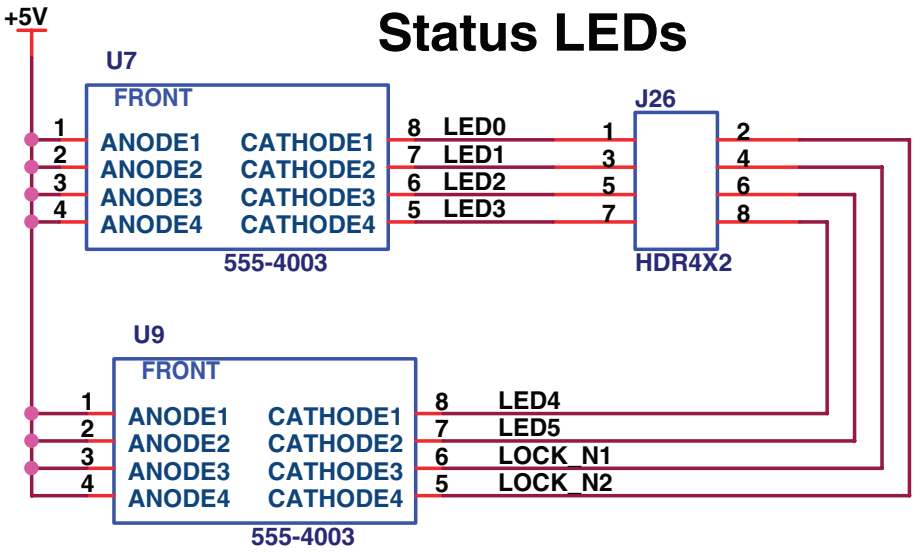


Figure 8-4 J26 LED Signals Header

The Pin Outs for J26 are shown in Table 8-1.

Table 8-1 J26 Pin Outs

Pin	Signal
1	LED0
2	LOCK_N2
3	LED1
4	LOCK_N1
5	LED2
6	LED5
7	LED3
8	LED4

Bus Bars

The two bus bars, **B1** and **B2** are installed to prevent flexing of the PWB and serve no other purpose. They are connected quite solidly into the ground plane of the DN3000k10S at every hole, and you can use the metal bars to ground-test equipment such as oscilloscopes and pattern generators. Be careful not to short any power rails or signals to these metal bars—they can carry a lot of current. The PCI bracket, **BRK1**, is also connected to the ground plane at each of the screw mounts.

The 200 Pin Connectors: J23, J24, J25

The DN3000k10S contains three 200-pin connectors, **J23**, **J24** and **J25**. Daughter cards of any sort may be plugged into these connectors. The relative pin location of the powers, grounds, and signals is identical for each of the three connectors, which means that the same daughter card can be plugged into any of the three slots. A hole that can be used to attach a standoff is located at the same relative position from each connector Figure 8-5. This hole is grounded on the DN3000k10S, so connect this mounting hole to digital ground on your daughter card.

The mechanical position of the 200-pin connectors on the DN3000k10S is shown in Figure 8-5. The 200-pin connector used on the DN3000k10S is a Berg Electronics 91294-003 in the Micropax™ family. This link will take you to the Berg website: **http://www.berg.com/**. This Berg connector was chosen because of its high pin density, performance, and availability. The part number for the mating connector is 91403-003. We stock the mating connector at our offices in La Jolla, CA, so if you are designing a daughter card and are having trouble getting this part, call us. We would be happy to send you a few at our cost. Appendix A contains a mechanical datasheet for both the Berg 91403-003 and 91294-003 connectors.

This style of connector has four mounting holes—two screw holes at each end and two alignment holes between pins 50–51 and after pin 100 (see Figure 8-5). These mounting holes are part of the metal shell of the connector and make an important connection to the mating connector. All four of these mounting holes are connected to digital ground on the DN3000k10S—therefore, the shell of the connector is grounded.

We used the pin numbering shown in Figure 8-5 for the 200-pin, 91294-003 connectors.

The Signals

Each of the three 200-pin connectors has the following:

- 162 signals connected to the FPGA
  - All 162 are connected to the FPGA
  - A subset of the 162 are also connected to the memories
- 7 clocks
- The following power rails:
  - +12V (1 pin)
  - -12V (1 pin)
  - +5V (2 pins)
  - +3.3V (2 pins)
  - +1.5V (2 pins)
  - GND (23 pins + case)

Regarding the amount of current that the power pins can carry, the following text is lifted directly from the specification for the Micropax™ family of connectors:

> *6.1 Current Rating. Current rating shall be evaluated in still air at 25°C ambient temperature. Under the following conditions, the temperature rise shall be no greater than 30°C:*
>
> *All contacts powered at 0.5 amp*
>
> *One contact powered at 3.0 amps*

Most of the signals are TTL (or some low current variation such as LVDS), so you can reasonably expect to get up to 3 amps per power pin through this connector. Remember that the +3.3V and +1.5V power supplies are limited to 5 amps total—the memories, the FPGA and the clock circuitry on the DN3000k10S consume +3.3V. The FPGA only consumes +1.5V.

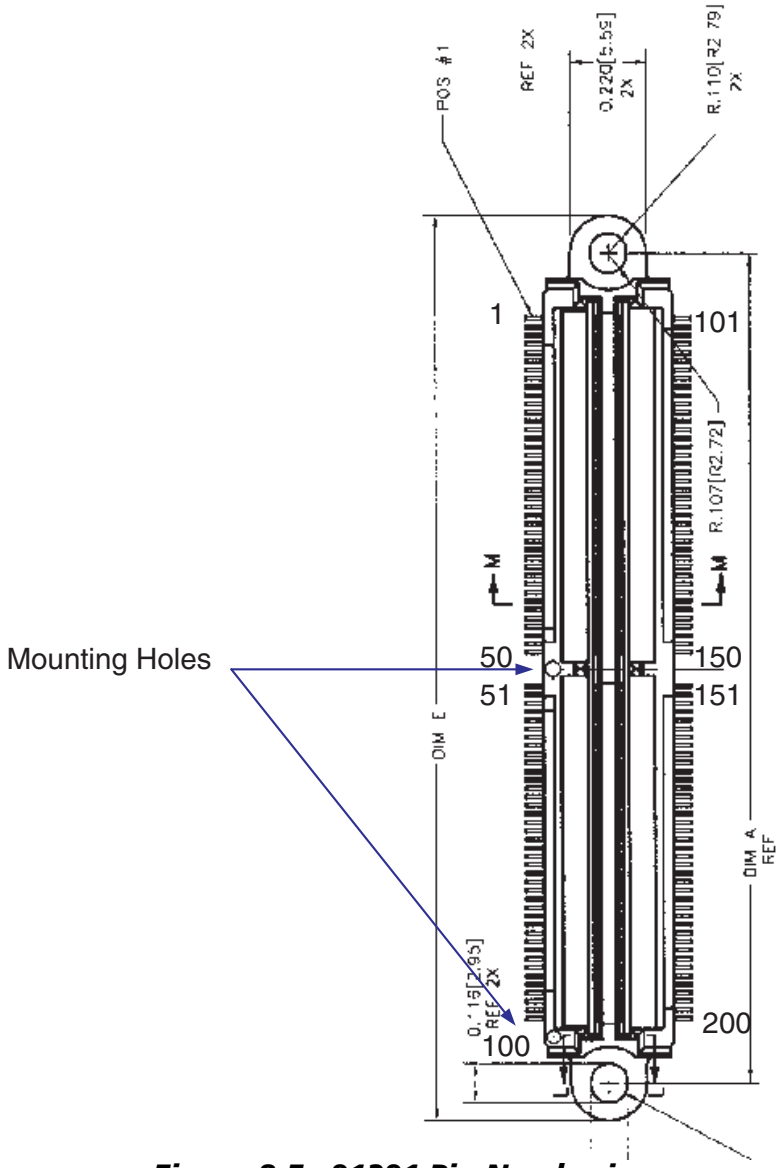


Figure 8-5 91291 Pin Numbering

If you need more power, consider using a cable connecting **J17** to your daughter card in addition to the pins on the 200-pin connectors. See "Header J17: Off-Board Power" on page 6–3. for more detail.

If you use the DN3000k10S stand-alone (meaning that it is *not* plugged into a PCI slot), the auxiliary power connector has +5V and +12V, but does not have –12V. So unless you provide –12V to the DN3000k10S via another connection, –12V will not be available for use by a daughter card.

NOTE: –12V is not required by the DN3000k10S. The DN3000k10S will operate normally without a –12V power supply.

Some of the interconnect with the 200-pin connectors is shared with the memories. See Figure 5-1 on page 5-2 to see which signals are shared. **J25** (**CON1**) has no signals that are shared (Figure 8-6).

Notes:

- Some signals have an "X" designation. Early in the design stage for the DN3000k10S, the "X" designation meant that the signal existed for the 2v4000/6000/8000 but not for the 2v3000. Signals were swapped during the torturous layout process, and in the final product, this is no longer true. So, signals with an "X" designation are no different than other FPGA signals.
- Be careful with the control signals for the SSRAMs that are shared with **J23** and **J24**. If the OE is active, the SSRAM drives its data bus. If the CEs are active, a write may occur.
- The signals designated "MBCK?" where "?" is 0, 1, 2, 3, 4, 5, 6, 7 or 8 are connected only between the 200-pin connectors and the FPGA. These signals are connected to a clock input pin on the FPGA, and therefore can be used as a method to externally clock the FPGA. Also, some of MBCK's may be combined to provide differential I/Os.

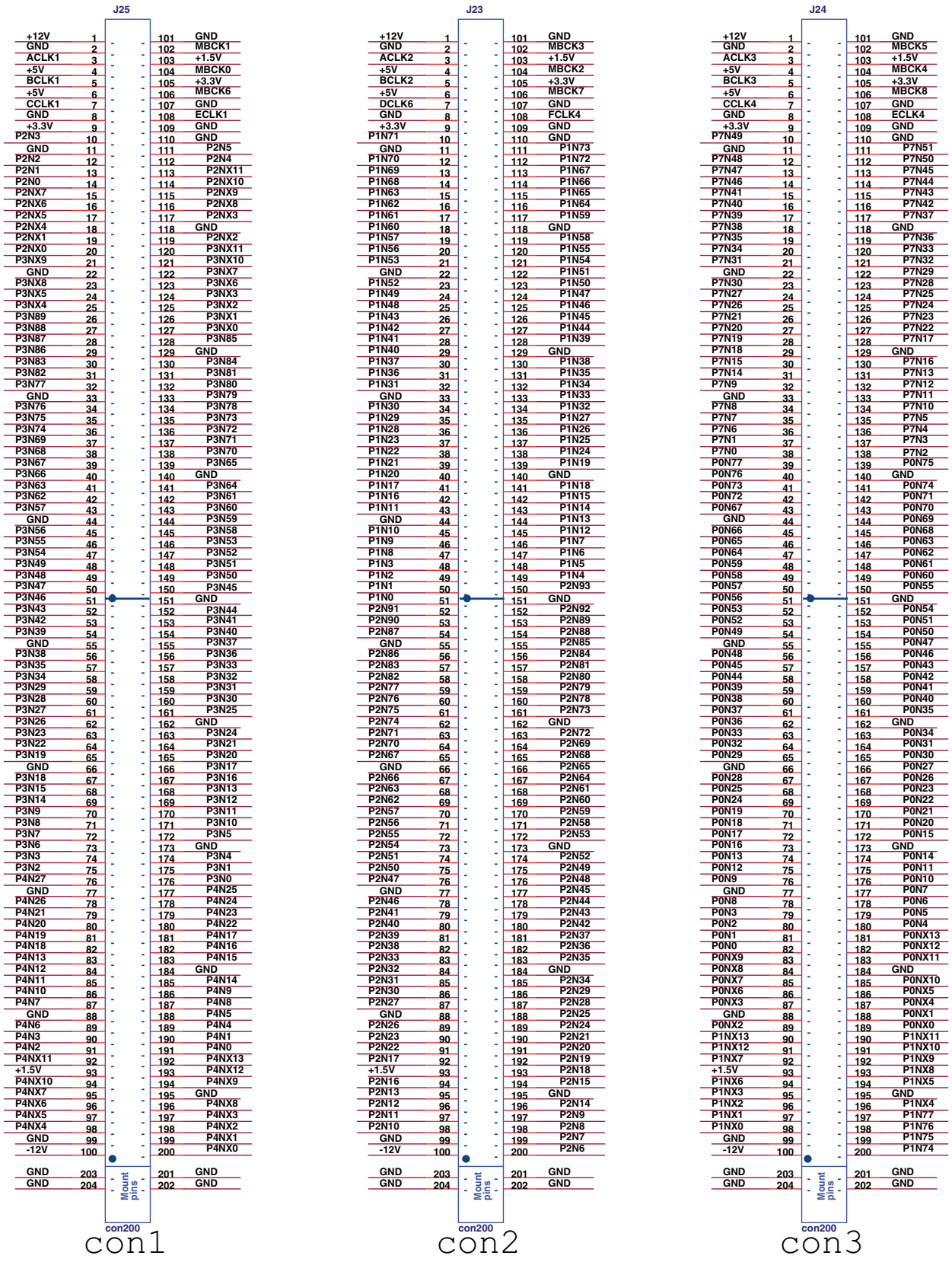


Figure 8-6 200 Pin Connectors — Signal Connections

Chapter 9

Utilities

PCI Debug—General Pontificating

Debugging of PCI-based hardware can be troublesome, so it is best to do so with a tiered approach. The following sequence of events needs to occur for a PCI-based peripheral to start working:

1. The hardware must boot itself at power-up — in the case of the DN3000k10S:
   a. The µP must boot.
   b. Recognize the SmartMedia card.
   c. Configure the FPGA. (Hopefully, all this occurs before RST# on the PCI bus is deasserted.)
2. The PCI BIOS executes the PNP routines and configures the BARs on all PCI peripherals.
3. The operating system driver initializes the card.
4. The application initiates communication with the driver and the application executes.

The steps are dependant. Each of the steps must start and execute flawlessly before the next step occurs. When you get a PCI card for the first time, it is necessary to debug each step before attempting to go to the next. We provide utilities to help with each step.

Steps 1 and 2 are best done without an operating system in place. Windows NT-based systems take minutes to reboot after a crash (the-BLUE-screen-death) and an NT driver won't work unless the hardware is debugged. Since crashing is a regular occurrence in a PCI hardware debug environment, we find it easiest to do our debug and manufacturing test in the old DOS environment. Virtually all PCI peripherals get configured with addresses beyond the IM boundary. On a PC, C programs cannot access memory locations beyond IM unless special programs called DOS extenders are used. Several freeware DOS extenders are available. We use a free DOS-extender called DJGPP. More information can be found at **http://www.delorie.com/DJGPP**.

PC-Based—AETEST.EXE

A utility program called AETEST is provided with the DN3000k10S. AETEST can be run under DOS, Windows 98/ME, Windows NT/2000, or LINUX. When used under DOS, you must boot your PC with a DOS disk. We ship one with the DN3000k10S in case you don't know how to make one on your own. All features work in the native mode of AETEST, which is DOS.

All source code for AETEST is provided, so you are welcome to customize the program to your own applications.

AETEST is not a stable program. We add and subtract features when we need to for debug and verification purposes, so don't be concerned if the screens that you see aren't exactly replicated here.

In a nutshell, AETEST lets you do the following:

- Determine if PCI recognizes the DN3000k10S
- Read/write/loop any memory location
- Read/write/loop configuration space
- Display all configured PCI devices
- Display memory setting from any locations
- Fill memory with various patterns
- Run various tests on the DN3000k10S
  - SSRAM Test
  - Multiplier Test
  - SDRAM Test
  - Interconnect Test
  - Daughter Card Test

AETEST Utility Installation Instructions

Installation Instructions for DOS

1. The files `aetestdj.exe` and `cwsdpmi.exe` (the DOS extender) need to be in the same directory.
2. Run `aetestdj.exe`.

Installation Instructions for Windows NT

1. Install the device driver: `install.exe` and `qldriver.sys` must be in the same directory.
2. Type "`install`"
3. After the driver is installed, start the driver by selecting **Control Panel–>Devices–>**find **"QLDriver"** –>click **"Start"**
4. Run `aetestnt.exe`.

Installation Instructions for Windows 2000

1. Install the device driver: `qldriver2000.inf` and the driver file (`qldriver.sys`) should be in the same directory.
2. Open **Control Panel**, click on "**Add/Remove Hardware**" and then go to "**Next–>**."
3. Choose **Add/Troubleshoot a device** (the default option) and click on "**Next–>**."
4. Wait until it finishes new hardware device searching; choose "**Add a new device,**" and click on "**Next–>**."
5. Choose "**No, I want to select the hardware from a list,**" and press "**Next–>**."
6. Choose "**Other Devices**" from **Hardware Types** list and press "Next–>"

7. Click on "**Have Disk…**"
8. In "**Copy Manufacturer's Files From:**" window, find the directory where `qldriver.sys` is located, then press "**OK**."
9. You should see "`dn2000k10 driver`" under **Models**; click on "**Next–>**."
10. Press "**Next–>**" and then "**Finish**."
11. Run `aetestnt.exe`.

Installation Instructions for LINUX

This has been tested on Red Hat Linus 7.2 (kernel version 2.4.x).

Note that all the text files, including the scripts, are DOS text format (with an extra carriage return character after every new line), so you need to convert them.

1. You must be root to start the driver and the program. "`dndev_load`" and "`dndev_unload`" are scripts that load and unload the driver; "`dndev.o`" is the driver file.
2. Load the driver; type "`sh dndev_load`"
3. Unload the driver; type "`sh dndev_unload`"
4. After driver is loaded, run the utility `aetest_linux`.
5. Note: You might need to run chmod on `aetest_linux` to make it executable: type "`chmod u+x aetest_linux`."

Installation Instructions for Solaris

The utility and driver are tested on Solaris 7.0/Sparc, with the 32-bit kernel.

Note that all the text files, including the scripts, are DOS text format (with an extra carriage return character after every new line), so you need to convert them.

1. To install the driver, go to the driver directory, make sure the driver file "`dndev`" is in the sparc sub-directory, and run "`sh dndev_uninstall.sh`"
2. To uninstall the driver, run "`sh dndev_uninstall.sh`"
3. To run the test utility, run "`aetest_solaris`" as root after the driver is loaded.

The driver is compiled with the gcc compiler.

`aetest_solaris` is compiled with "gmake." You can download it from the GNU website. The "make" from the Solaris installation does not work with our makefile format.

You may need to make `aetest_solaris` executable; run "`chmod u+x aetest_solaris`."

Installation Instructions for Windows 98/ME

There are two ways to run AETEST: You can run the DOS version "aetestdj.exe" directly, or you can run AETEST with a device driver.

To run AETEST with a device driver follow the steps below.

1. Choose a default PCI driver for the device. When Windows first starts with the device plugged in, it should ask for a device driver. Select **"Specify the location of the driver."**
2. Select **"Display a list of the drivers in a specifice location…"**
3. Select **"Other devices."**
4. Under **"Manufacturers"** tab, select **"unknown device**."
5. Under "Models" select **"unsupported device**."
6. The driver file (pcifg.vxd) and aetest98.exe must be in the same directory. Run aetest98.exe.

NOTE: To re-compile the driver file pcicfg.vxd, you need the VtoolsD compiler from **www.numega.com**.

AETEST Options: Description and Definitions

Startup

When AETEST is first started, it tries to find a device that it recognizes. We have arbitrarily defined the DN3000k10S with a DEVICE_ID of 0x1240 and a VENDOR_ID of 0xABCD. You should see the following screen if AETEST recognizes a DN3000k10S (Figure 9-1):

```
searching for "DN2000K10 Asic Emulator (1000E)" VENDOR_ID==abcd, DEVICE_ID==1236

searching for "DN2000K10 Asic Emulator (1600E)" VENDOR_ID==abcd, DEVICE_ID==1237

searching for "DN3000K10S Asic Emulator (6000)" VENDOR_ID==abcd, DEVICE_ID==1240

found device ---- vabcd, d1240 name="DN3000K10S Asic Emulator (6000)"
Configuration space:
00: 1240abcd      04: 0000001f
08: ff000047      0c: 00000000
10: fd800000      14: e0000000
18: 00000000      1c: 00000000
20: 00000000      24: 00000000
28: 00000000      2c: 90ab5678
30: 00000000      34: 00000000
38: 00000000      3c: 00000000
BAR0: base: 0xfd800000, size: 0x00800000
BAR1: base: 0xe0000000, size: 0x10000000
BAR2: base: 0x00000000, size: 0x00000000
BAR3: base: 0x00000000, size: 0x00000000
BAR4: base: 0x00000000, size: 0x00000000
BAR5: base: 0x00000000, size: 0x00000000
press any key
```

Figure 9-1 AETEST Startup Screen, DN3000k10S Recognized

Most of this initial display is debug information. The program is looking for a Vendor and Device ID that it recognizes, and finds vendor=abcd and device=1240, which is a DN3000k10S stuffed with a 2V60000. The

lines after `Configuration space:` show what is in the configuration space and how the BARs are configured.

If AETEST does **not** see a PCI peripheral it recognizes, you will see the following (Figure 9-2):

```
searching for "Quad Sharc" VENDOR_ID==5045, DEVICE_ID==1
searching for "DN2000K10 Asic Emulator" VENDOR_ID==abcd, DEVICE_ID==1234
searching for "DN2000K10 Asic Emulator (2000E)" VENDOR_ID==abcd, DEVICE_ID==1235
searching for "DN2000K10 Asic Emulator (1000E)" VENDOR_ID==abcd, DEVICE_ID==1236
searching for "DN2000K10 Asic Emulator (1600E)" VENDOR_ID==abcd, DEVICE_ID==1237
searching for "DN3000K10S Asic Emulator (6000)" VENDOR_ID==abcd, DEVICE_ID==1240
searching for "ql5064 interface test" VENDOR_ID==1234, DEVICE_ID==5678
searching for "ql5064 64MB dram+LFSR" VENDOR_ID==1234, DEVICE_ID==5679
searching for "ql5064 PowerPC bridge" VENDOR_ID==11e3, DEVICE_ID==6
searching for "ql5064 PowerPC bridge (old VID/DID)" VENDOR_ID==1010, DEVICE_ID==5064
searching for "ql5064 AntiFuse test #1" VENDOR_ID==71f3, DEVICE_ID==2454
searching for "ql5064 AntiFuse test #2" VENDOR_ID==507, DEVICE_ID==2367
searching for "ql5064 AntiFuse test #3" VENDOR_ID==bc92, DEVICE_ID==2e6c
searching for "ql5064 AntiFuse test #4" VENDOR_ID==e125, DEVICE_ID==c38c
searching for "ql5064 AntiFuse test #5" VENDOR_ID==e62c, DEVICE_ID==ca76
searching for "ql5064 AntiFuse test #6" VENDOR_ID==448b, DEVICE_ID==e6a
searching for "ql5064 Emulated with 8051" VENDOR_ID==1243, DEVICE_ID==4321
searching for "Greg's PCI Device" VENDOR_ID==5143, DEVICE_ID==2
searching for "Cohu's PCI Device (sensor board)" VENDOR_ID==dead, DEVICE_ID==beef
searching for "LYNX 9610" VENDOR_ID==10b5, DEVICE_ID==9610

Didn't find known device in the following list:
   vendor_id=5045, device_id=1
   vendor_id=abcd, device_id=1234
   vendor_id=abcd, device_id=1235
   vendor_id=abcd, device_id=1236
   vendor_id=abcd, device_id=1237
   vendor_id=abcd, device_id=1240
   vendor_id=1234, device_id=5678
   vendor_id=1234, device_id=5679
   vendor_id=11e3, device_id=6
   vendor_id=1010, device_id=5064
   vendor_id=71f3, device_id=2454
   vendor_id=507, device_id=2367
   vendor_id=bc92, device_id=2e6c
   vendor_id=e125, device_id=c38c
   vendor_id=e62c, device_id=ca76
   vendor_id=448b, device_id=e6a
   vendor_id=1243, device_id=4321
   vendor_id=5143, device_id=2
   vendor_id=dead, device_id=beef
   vendor_id=10b5, device_id=9610

Hit a key to continue
```

Figure 9-2 AETEST Startup Screen, No PCI Peripheral Recognized

AETEST will still run, but many DINI product-specific options will not be available.

AETEST Main Screen

The AETEST Main Screen is shown in Figure 9-3.

```
            -=-= ASIC Emulator PCI Controller Driver =-=- v8

                    0) Read FPGA revision
                    1) PCI Menu
                    2) Memory Menu
                    3) Flash Menu
                    4) Clock Menu
                    5) Dedicated Multiplier Test

                    Q) Quit

                     -=-= PCI BASE ADDRESS =-=-
          0 : fd800000    1 : e0000000    2 : 00000000
          3 : 00000000    4 : 00000000    5 : 00000000



    Please select option:
```

Figure 9-3 AETEST Main Screen

Options

Read FPGA Revision. Display the revision ID of the FPGA. We will update the revision ID of the FPGA every time we change the reference design.

PCI Menu. Display the PCI utilities menu.

Memory Menu. Display the Memory Menu.

Flash Menu. Display the FLASH Utilities Menu (DN2000k10 series only!).

Clock Menu. Display the Clock Utilities Menu.

Dedicated Multiplier Test. Execute the multiplier test.

Q. Quit and return to the DOS prompt

The selections are sometimes case sensitive, so be aware of the status of the CAPS LOCK on your keyboard. The base addresses for each of the configured BARs is displayed on all screens. You will need these addresses if you want to manually read and write to address locations within the PCI reference design. In this example (Figure 9-3, above), BAR0 is configured to `0xFD800000` and BAR1 is configured to `0xE0000000`. BAR[5:2] are not configured so they show up as `0x0`.

PCI Menu

The AETEST PCI menu is shown in Figure 9-4.

```
          -=-= ASIC Emulator PCI Controller Driver =-=- v8
      PCI Device/Function Num: 0x7F, 0x00

S) Set PCI Device Number
F) Set PCI Function Number
D) Display all Configured PCI Devices
1) Display Vendor and Device ID for PCI device-function: 7f-0
2) Loop on PCI device-fun: 7f-0 and Display Vendor and Device ID
3) Loop on PCI device-fun: 7f-0 and Don't Display Vendor and Device ID
4) Loop on all PCI device numbers and Display Device/Vendor ID's
5) Display all PCI information for PCI device-function: 7f-0
6) Write config dword
7) Read config dword

C) Configure BAR's from File
V) Save BAR Configuration to File
M) Main Menu
Q) Quit

                -=-= PCI BASE ADDRESS =-=-
        0 : fd800000    1 : e0000000    2 : 00000000
        3 : 00000000    4 : 00000000    5 : 00000000

Please select option:
```

Figure 9-4 AETEST PCI Menu

Set PCI Device Number. Sets a PCI device number of your choice as the "active" device (hex input). This option lists the available Device Numbers to help you match up your Device ID and Vendor ID with the device number.

Set PCI Function Number. Sets a PCI function number of your choice as the "active" function of a multi-function device (hex input). This option lists the Device ID and Vendor ID of each function within the "active" device number to help you to choose the desired function.

Display all Configured PCI Devices. Displays the PCI Device Numbers and corresponding Device ID and Vendor ID of all devices seen on the bus. This does not display device numbers with a Device ID and Vendor ID of all ones (`OxFFFF`).

Display Vendor and Device ID for PCI device-function.

Displays the Vendor ID and Device ID of the active device and function number. In the example above, this would display the Vendor ID and Device ID of the PCI device at device number `0x7F`, function number `0x00`.

Loop on PCI device-fun: 7f-0 and Display Vendor and Device ID. Reads and displays the Vendor ID and Device ID of the "active" device number and function number. Repeats this action until the user hits a key to stop it.

Loop on PCI device-fun: 7f-0 and Don't Display Vendor and Device ID. Same as previous menu option, except doesn't display results. This menu option is useful when using an oscilloscope to debug configuration reads.

Loop on all PCI device numbers and Display Device/Vendor ID's. Loops on each device number, reading the Vendor ID and Device ID for each. It moves onto the next device number when you press any key. That is, it continually reads the Vendor ID and Device ID from device number 0 until you hit a key, at which point it continually reads the Vendor ID and Device ID from device number 1. It moves all the way through device number 0 to device number `0x7F` (in case there are any bridges on your PCI bus).

Display all PCI information for PCI device-function: 7f-0.

Reads and displays all of the configuration space for the "active" device and function number. Use options "S" and "F" to change between the "active" device number and function number, and then use this option to view the entire configuration space.

Write config(uration) DWORD. Allows write to configuration space. The following text will appear to remind you what is in configuration space for a PCI device:

```
PCI_CS_VENDOR_ID                0x00
PCI_CS_DEVICE_ID                0x02
PCI_CS_COMMAND                  0x04
PCI_CS_STATUS                   0x06
PCI_CS_REVISION_ID              0x08
PCI_CS_CLASS_CODE               0x09
PCI_CS_CACHE_LINE_SIZE          0x0c
PCI_CS_MASTER_LATENCY           0x0d
PCI_CS_HEADER_TYPE              0x0e
PCI_CS_BIST                     0x0f
PCI_CS_BASE_ADDRESS_0           0x10
PCI_CS_BASE_ADDRESS_1           0x14
PCI_CS_BASE_ADDRESS_2           0x18
PCI_CS_BASE_ADDRESS_3           0x1c
PCI_CS_BASE_ADDRESS_4           0x20
PCI_CS_BASE_ADDRESS_5           0x24
PCI_CS_EXPANSION_ROM            0x30
PCI_CS_INTERRUPT_LINE           0x3c
PCI_CS_INTERRUPT_PIN            0x3d
PCI_CS_MIN_GNT                  0x3e
PCI_CS_MAX_LAT                  0x3f

Input config offset (hex 0x00-0xff):
word to write (in hex):

Loop indefinitely? (y or n)?
```

If looping was selected, any keypress will stop the loop.

Read config(uration) DWORD. Allows read from configuration space. Has options for single read, loop read with display, and loop read without display.

Configure BARs from File. Reloads the PCI configuration of the "active" device from a file. It writes 0x001F to the command register, and writes the 6 bars with the values from the file. This is useful for hot-swapping devices (power switch still required on extender), or reinitializing a device when its configuration has been altered.

WARNING: Because the PCI BIOS is not assigning the BARs for this device, you may induce a memory conflict by using this option. This option is for advanced users only!

Save Bar Configuration to File. Writes PCI Device ID, Vendor ID and the BARs into a file (from the "active" device). This option is for advanced users only!

Memory Menu

The memory menu (Figure 9-5) allows you to perform a variety of tests of PCI memory along with some DN3000k10 specific tasks.

```
 -=-= ASIC Emulator PCI Controller Driver =-=- v8

1) Write To Memory Test        2) Read Memory Test
3) Write/Read Test             4) Memory Fill
8) Memory Display
9) Write Memory Byte
a) Read Memory Byte
b) Write/Read Memory Byte
c) memory test on SSRAM 1
d) memory test on SSRAM 2
e) memory test on SSRAM 3
f) memory test on SDRAM
g) full memory test (including blockram)
n) memory test on FPGA block memory
p) bar memory range test
u) SRAM memory test
M) Main Menu                   Q) Quit

          -=-= PCI BASE ADDRESS =-=-
0 : fd800000    1 : e0000000    2 : 00000000
3 : 00000000    4 : 00000000    5 : 00000000
```

Figure 9-5 AETEST Memory Menu

Write to Memory Test. Write a selected number of long words to a specific PCI memory location (Figure 9-6).

```
                          --------
Memory location (hex)     :fb000000
Numbers of long words to write (in decimal) ?  2
long word to write (in hex) :aaaaaaaa
long word to write (in hex) :55555555

Loop indefinitely? (y or n)?

Hit a key to continue....
```

Figure 9-6 AETEST Write to Memory Test

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be accessed. A minimum of 1 to a maximum of 1024 long words can be written, in sequential order, to the same address. A looping option is available if you want to use an oscilloscope. If you are in a scope loop, any keypress will terminate the loop and return you to the main menu.

Read Memory Test. Read a single long word from a specific PCI memory location (Figure 9-7).

```
              --------Input address :fb000000

fb000000
        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-7 AETEST Read Memory Test

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be read. Three options are available:

1. Read once and display.
2. Read indefinitely and display.
3. Read indefinitely and don’t display.

Write/Read Test. Write a long word to a specific PCI memory location and immediately read what was written. Repeat for a selected number of long words (Figure 9-8).

```
Input address :fb000000

Numbers of long words to write (in decimal) ?  2

long word to write (in hex) :000
long word to write (in hex) :aaa

        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-8 AETEST Write/Read Test

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be read. The program will prompt for the number of long words you with to write (1 to 1024). Three options are available:

1. Read once and display.
2. Read indefinitely and display.
3. Read indefinitely and don't display.

Option 3 is a very useful scope loop.

Memory Fill. Fill memory with a selected pattern (Figure 9-9).

```
Input starting address (hex and 32 bit aligned): fb000000

Input number of bytes (divisible by 4): 1000
1 -- Fill with 0
2 -- address=data
3 -- 0x55555555, 0xaaaaaaaa
4 -- 0xffffffff
5 -- data=~address
```

Figure 9-9 AETEST Memory Fill

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be written. The program will prompt for the number of bytes (in hex) you wish to fill (`4` to `0xfffffffc`). The following fill options are available:

1. **fill with 0** — fill all the locations with `0x00000000` (clear the memory)
2. **address=data** — fill each long word with its address
3. alternating `0x55555555`, `0xAAAAAAAA`
4. `0xffffffff` — set all of memory
5. **data=~address** — fill each long word with the address (each bit inverted).

Memory Display. Display 160 long words of memory. You are prompted for the starting address (in hex):

```
Input starting address (hex and 32 bit aligned):
```

The following screen is displayed (Figure 9-10):

```
          0        4        8        c       10       14       18       1c
000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000120 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000140 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000180 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000200 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000220 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000240 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000260 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
(f)orward  (b)ack  (j)ump  goto(0)  (q)uit
```

Figure 9-10 AETEST Memory Display

(f) forward — pages the screen forward in memory.
(b) back — pages the screen backwards in memory.
(j) jump — jump to a specific location (in hex).
(0) goto — jump back to the original address location specified at the beginning.
(d) delay and display loop — display, wait for a second, and display again. Loop until a key is struck.

Write Memory Byte. Write a specific number of bytes to a single memory address (Figure 9-11).

```
Input address :fb000000

Numbers of long words to write (in decimal) ?  2

byte to write (in hex) :ff
byte to write (in hex) :aa

        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-11 AETEST Write Memory Byte

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be accessed. A minimum of 1 to a maximum of 1024 bytes will be written, in sequential order, to the same address. A looping option is available if you want to use

an oscilloscope. If you are in a scope loop, any keypress will terminate the loop and return you to the main menu.

Read Memory Byte. Read a single byte from a specific PCI memory location (Figure 9-12).

```
Input address :fb000000

fb000000
        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-12 AETEST Read Memory Byte

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be read. Three options are available:

1. Read once and display.
2. Read indefinitely and display.
3. Read indefinitely and don't display.

Write/Read Memory Byte. Write and read a single DWORD from a specific PCI memory location. After entering a memory address (hex, 32 bits), you specify how many DWORDS you want written and read back, and the data. Then, you choose from the 3 options as above. The menu option does not perform any data checking. (Figure 9-13)

```
Numbers of long words to write (in decimal) ?  2

byte to write (in hex) :88888888
byte to write (in hex) :99999999

        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-13 AETEST Write/Read Memory Byte

Memory test on SSRAM1. Tests one of the SSRAM chips on the DN3000k10.

Memory test on SSRAM2. Tests one of the SSRAM chips on the DN3000k10.

Memory test on SSRAM3. Tests one of the SSRAM chips on the DN3000k10.

Memory test on SDRAM. Tests the SDRAM chip on the DN3000k10.

Full Memory Test (Including BlockRAM). Tests all of the memories. This includes the SSRAM chips, the SDRAM, and the BlockRAM internal to the FPGA.

Memory test on FPGA block memory. Tests the BlockRAM inside the FPGA. On the DN2000k10, the BlockRAM is only in FPGA F.

BAR memory range test. Generic memory test that prompts the user for BAR number, starting address offset, DWORD count, and number of iterations. The user is also prompted if the program should stop if error occurs, or if the program should display any errors that occur. This allows for maximum flexibility when debugging a design with an oscilloscope, or debugging any memories or memory locations on your PCI bus. The memory test is very complete, performing a write then a read to every location, a read from every location, and then a read/write/read test to every location. All other memory test options listed in the memory menu are based on this generic memory test function.

Appendix A

Berg Connector Datasheets

Figure A-1 and Figure A-2 contain the schematics for the Berg 91403-003 Connector.

Figure A-3 through Figure A-5 contain the schematics for the Berg 91294-003 connector.

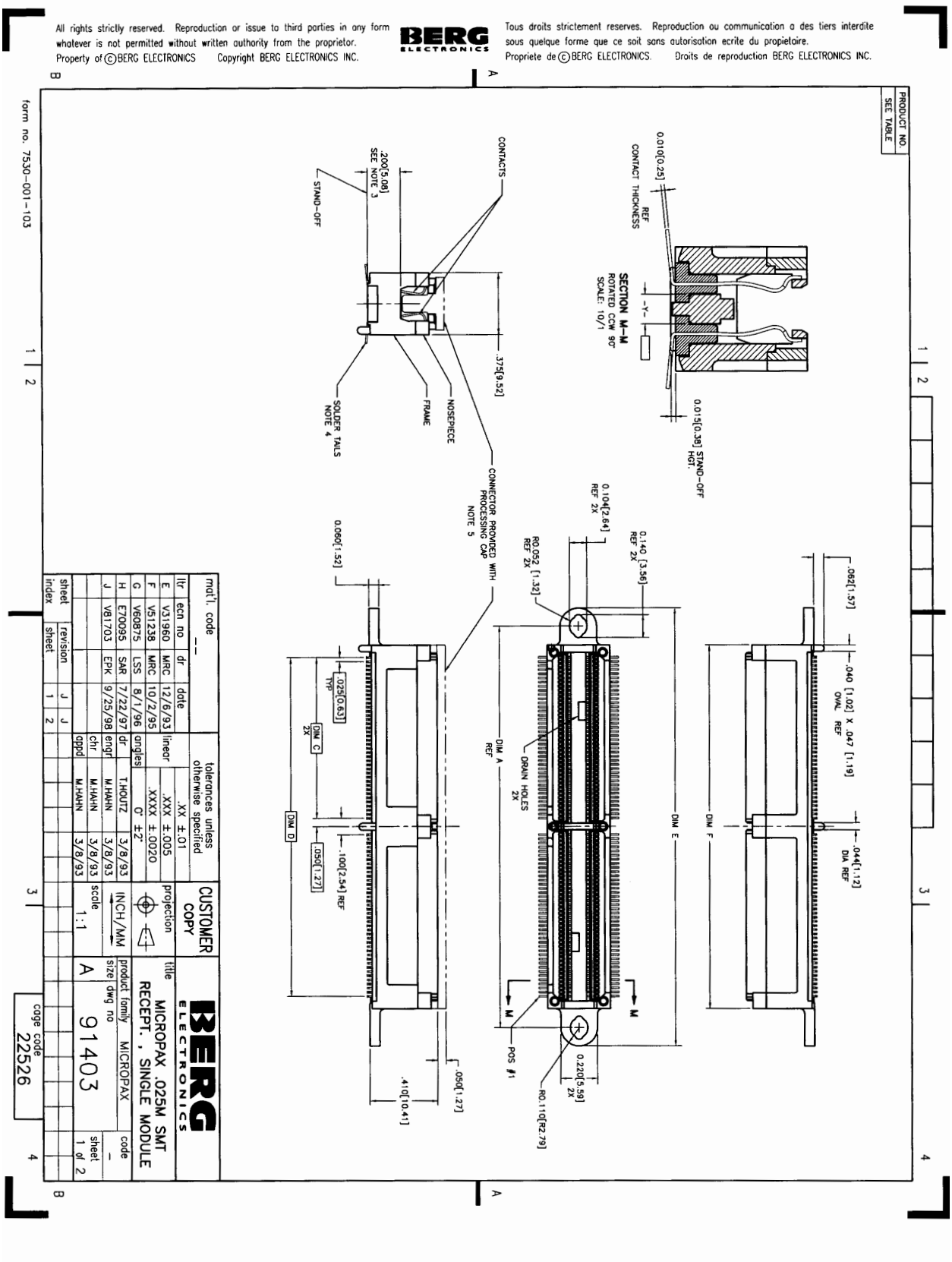


Figure A-1 Berg 91403-003 Datasheet Page 1 of 2

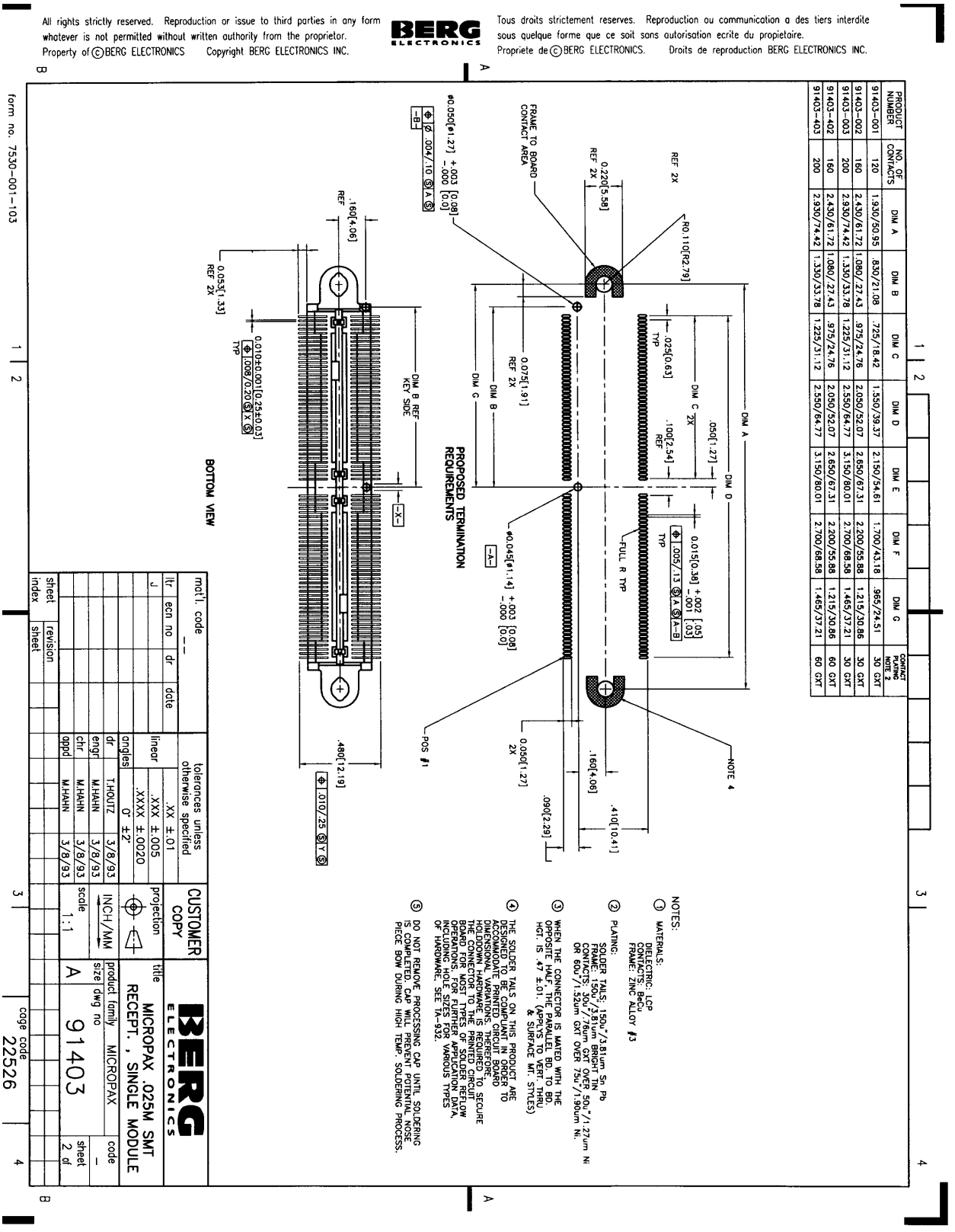


Figure A-2 Berg 91403-003 Datasheet Page 2 of 2

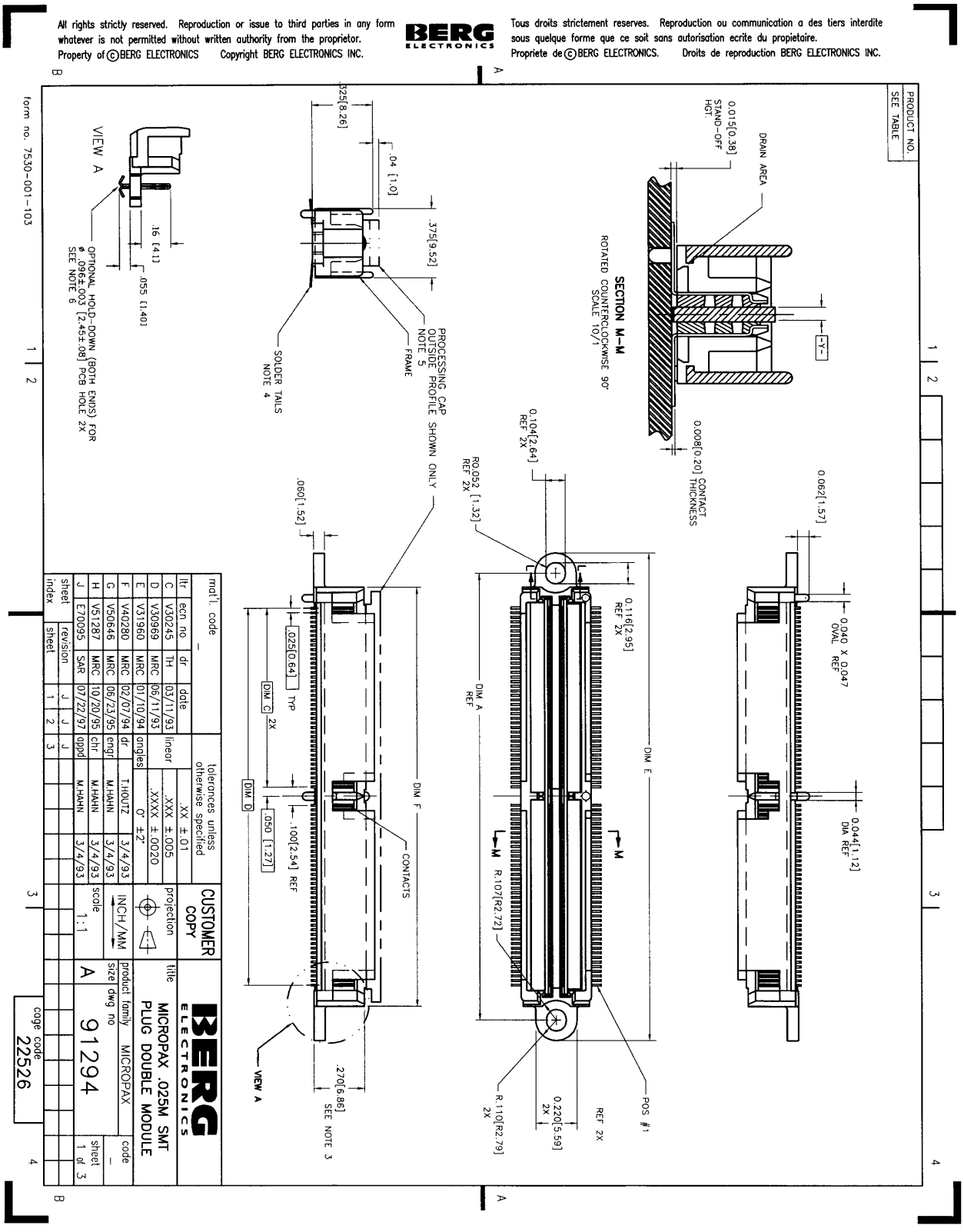


Figure A-3 Berg 91294-003 Datasheet Page 1 of 3

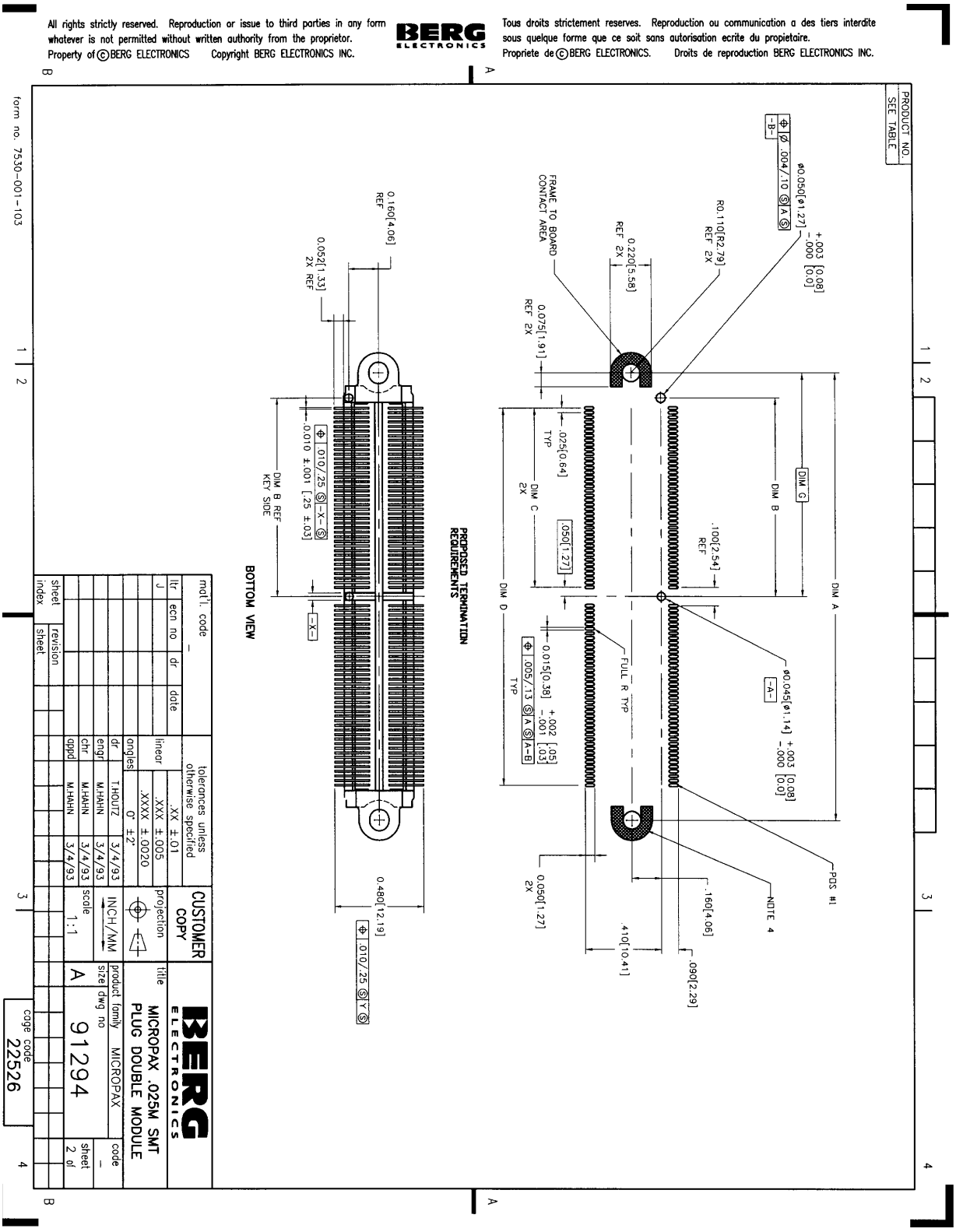
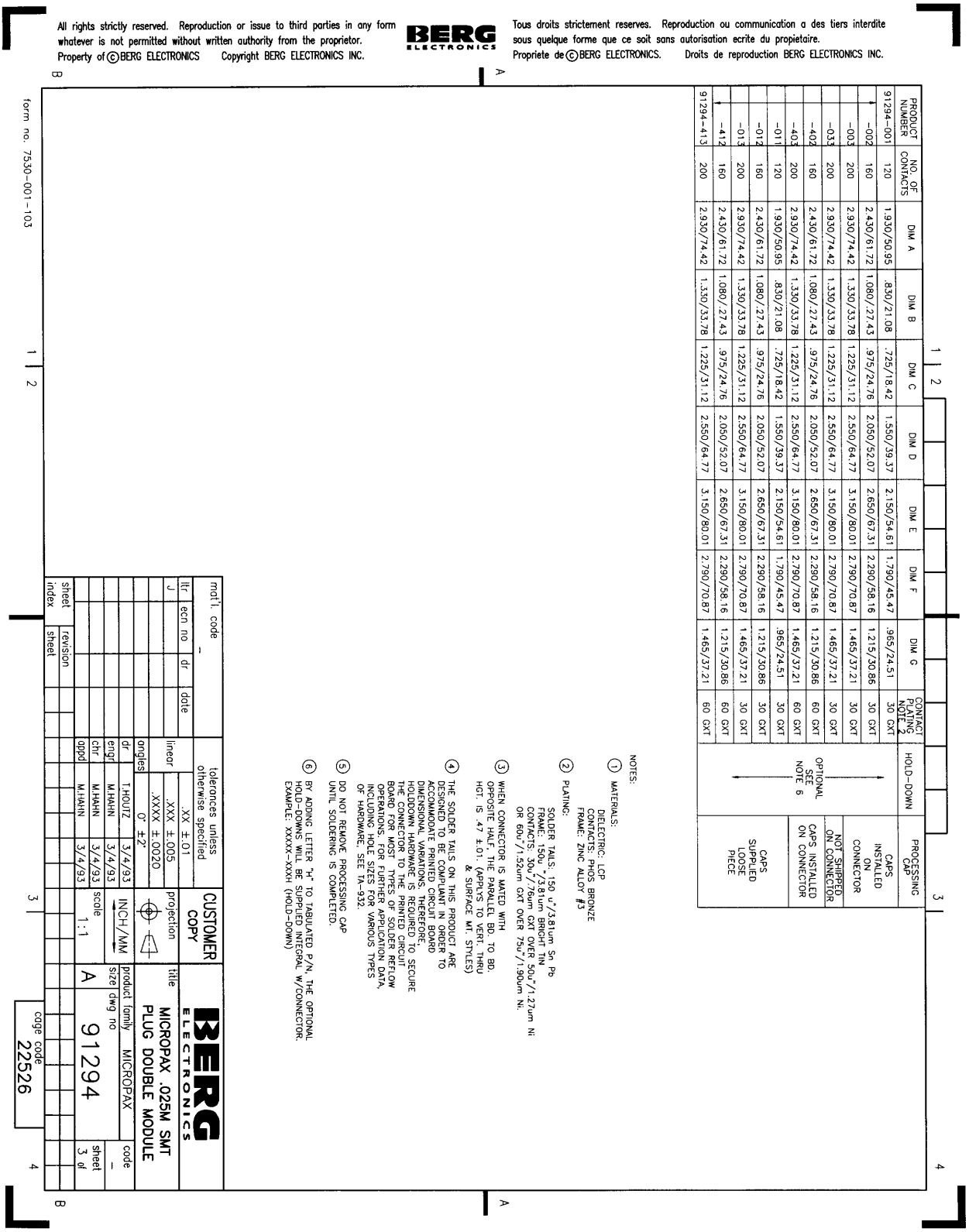


Figure A-4 Berg 91294-003 Datasheet Page 2 of 3



PRODUCT NUMBER	NO. OF CONTACTS	DIM A	DIM B	DIM C	DIM D	DIM E	DIM F	DIM G	CONTACT PLATING NOTE 2	HOLD-DOWN	PROCESSING CAP
91294-001	120	1.930/50.95	.830/21.08	.725/18.42	1.550/39.37	2.150/54.61	1.790/45.47	.965/24.51	30 GXT	OPTIONAL SEE NOTE 6	CAPS INSTALLED ON CONNECTOR
-002	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	30 GXT		
-003	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	30 GXT		
-033	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	30 GXT		NOT SHIPPED ON CONNECTOR
-402	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	60 GXT		CAPS INSTALLED ON CONNECTOR
-403	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	60 GXT		
-011	120	1.930/50.95	.830/21.08	.725/18.42	1.550/39.37	2.150/54.61	1.790/45.47	.965/24.51	30 GXT		
-012	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	30 GXT		CAPS SUPPLIED LOOSE PIECE
-013	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	30 GXT		
-412	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	60 GXT		
91294-413	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	60 GXT		

NOTES:

1. MATERIALS:
   DIELECTRIC: LCP
   CONTACTS: PHOS BRONZE
   FRAME: ZINC ALLOY #3
2. PLATING:
   SOLDER TAILS: 150 u"/3.81um Sn Pb
   FRAME: 150u"/3.81um BRIGHT TIN
   CONTACTS: 30u"/.76um GXT OVER 50u"/1.27um Ni
   OR 60u"/1.52um GXT OVER 75u"/1.90um Ni.
3. WHEN CONNECTOR IS MATED WITH OPPOSITE HALF, THE PARALLEL BD. TO BD. HGT. IS .47 ±.01. (APPLYS TO VERT. THRU & SURFACE MT. STYLES)
4. THE SOLDER TAILS ON THIS PRODUCT ARE DESIGNED TO BE COMPLIANT IN ORDER TO ACCOMMODATE PRINTED CIRCUIT BOARD DIMENSIONAL VARIATIONS. THEREFORE, HOLDDOWN HARDWARE IS REQUIRED TO SECURE THE CONNECTOR TO THE PRINTED CIRCUIT BOARD FOR MOST TYPES OF SOLDER REFLOW OPERATIONS. FOR FURTHER APPLICATION DATA, INCLUDING HOLE SIZES FOR VARIOUS TYPES OF HARDWARE, SEE TA-932.
5. DO NOT REMOVE PROCESSING CAP UNTIL SOLDERING IS COMPLETED.
6. BY ADDING LETTER "H" TO TABULATED P/N, THE OPTIONAL HOLD-DOWNS WILL BE SUPPLIED INTEGRAL W/CONNECTOR.
   EXAMPLE: XXXXX-XXXH (HOLD-DOWN)

form no. 7530-001-103

mat'l code –

ltr	ecn no	dr	date
J			

sheet index | revision sheet

tolerances unless otherwise specified
linear .XX ±.01 / .XXX ±.005 / .XXXX ±.0020
angles 0° ±2°

	dr	engr	chr	oppd
	T.HOLTZ	M.HAHN	M.HAHN	M.HAHN
	3/4/93	3/4/93	3/4/93	3/4/93

CUSTOMER COPY

projection

scale 1:1

INCH/MM

BERG ELECTRONICS

title MICROPAX .025M SMT PLUG DOUBLE MODULE

product family MICROPAX

size A | dwg no 91294 | code –

cage code 22526

sheet 3 of 3

Figure A-5 Berg 91294-003 Datasheet Page 3 of 3

Glossary and Acronyms

µP	microprocessor
A/D	analog to digital
AREF	Analog Voltage Reference
ASIC	application specific integrated circuits
AVCC	Analog Vcc
BAR	Base Address Register
BGA	ball grid array
BIOS	Basic Input/Output Services
CLB	configurable logic block
CMOS	complementary metal-oxide semiconductor
DCI	digitally controlled impedance
DCM	Digital Clock Manager
DES	data encryption standard
DIMM	dual in line memory module
EEPROM	Electrically Erasable PROM
EIA	Electronic Industries Association
ESD	electrostatic discharge
FAQ	frequently asked questions
FAT	file allocation table
FIFO	first in first out
FPGA	field programmable gate array
FT	flowthrough
GND	ground
HDL	Hardware Description Language
I/O	input/output
IDC	integrated desktop connector
IP	intellectual property
LED	light emitting diode
LSI	large scale integration
LUT	lookup table
LVCMOS	low voltage complementary metal-oxide semiconductor
LVDS	low voltage differential signalling
LVDS	Low-Voltage Differential Signaling
LVTTL	low voltage transistor-transistor logic
MDR	Mini D Ribbon
NIST	National Institute of Standards and Technology
PCI	peripheral component interconnect
PCI-X	peripheral component interconnect (extended)
PGP	Pretty Good Privacy
PL	pipelined
PLL	phase lock loop
PNP	plug-and-play
PWB	printed wire board
RAM	random access memory
RISC	reduced instruction set computer
RSA	A public-key cryptosystem developed by MIT professors Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman
RTL	resistor transistor logic
SDRAM	synchronous dynamic random access memory
SRAM	shadow random access memory
SSRAM	synchronous static random access memory
TDEA	riple data encryption algorithm
TTL	transistor-transistor logic

UCF	**user configuration file**
VHDL	**VHSIC Hardware Description Language**
VREF	**reference voltage**
ZBT	**zero-bus-turnaround**

Index

Symbols

A

B

C

Index (Continued)

D

E

F

Index (Continued)

Index (Continued)

R

S

Index (Continued)

Index (Continued)

T

Index (Continued)

U

V

W

X

Index (Continued)